INTERNATIONAL
HELLENIC
UNIVERSITY

Road obstacle recognition using Deep Learning models

Klearchos Stavrothanasopoulos

SID: 3308170023

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

DECEMBER 2018

THESSALONIKI – GREECE

INTERNATIONAL
HELLENIC
UNIVERSITY

Road obstacle recognition using Deep Learning models

Klearchos Stavrothanasopoulos

SID: 3308170023

Supervisor:	Prof. Konstantinos Diamantaras
Supervising Committee Members:	Dr. Christos Berberidis
	Dr. Agamemnon Baltagiannis

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

DECEMBER 2018

THESSALONIKI – GREECE

Abstract

Over the last decade researchers have made serious attempts to approach the Autonomous Driving (AD) vision. In the early stages of these attempts many machine learning and deep learning algorithms have been utilized without noteworthy success. Unfortunately, their slow detection speed was limiting their potential in driving conditions. Lately, some new papers were published proposing Convolutional Neural Network (CNN) based models, designed specifically for real-time object detection, most of them only trained and tested in general content datasets though. This research was aimed at exploring in detail three such systems (SqueezeDet, Yolo version 2, Yolo version 3) and testing their capability in the driving scene. We analyzed their performance on various cases using the KITTI 2D object detection dataset, one of the most representative datasets for autonomous driving. Besides KITTI, which was our main training and testing dataset, ImageNet and Pascal VOC 2007/2012 have been also used for the pre-training stage of the models.

Klearchos Stavrorthanasopoulos

December 2018

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisor Dr. Konstantinos Diamantaras for giving me the chance to explore such an interesting field of research and providing help and advice while conducting my thesis project. Also, I would like to thank him for his kind support throughout my MSc studies at the International Hellenic University.

In addition, I want to thank my family and close friends for being supportive in everything I pursue all these years, for believing in me and for strengthening my desire for continuous learning.

Abbreviations and Acronyms

AP Average Precision

BoW Bag of Words

CNN Convolutional Neural Network

CPU Central Processing Unit

FC Fully Connected (layer or network)

FCN Fully Convolutional Network

GPU Graphics Processing Unit

IoU Intersection over Union

mAP mean Average Precision

R-CNN Convolutional Neural Network with Region proposals

ReLU Rectified Linear Unit

RoI Region of Interest

RPN Region Proposal Network

SVM Support Vector Machine

YOLO You Only Look Once

YOLOv1 You Only Look Once version 1

YOLOv2 You Only Look Once version 2

Table of Contents

1 Introduction

Humans, ever since their evolution, are searching for ways to either minimize errors or eradicate them completely. This quest is nowadays ever increasing and has been focused on the prevention of accidents and especially road accidents which are constantly multiplying. Through precision and proactive measures, they could be eliminated to a large extent if not completely. As part of the solution, integrated machines have been introduced to aid scientists in their research for autonomous vehicles and assisted driving. In fact, machines have been actively established in many areas to assist humans and perform their tasks. This has been the trend so far and will be the trend in future. More research is being conducted by scientists and technical experts to make machines part of the modern life and this is where the Machine Learning plays an important part. Machine Learning can infest intelligence in the machines, and make them fully automated, error-free, and precise. Deep Learning, one of the subfields of machine learning, is based on certain models using artificial neural networks which are inspired from the functions, as well as the structure of the human brain. In simpler words, with the help of Deep Learning machines can simulate human brain and perform tasks with the required precision.

1.1 Inspiration for the research

This project has been inspired by the thriving progress of self - driving industry in the past few years. Automobile manufacturers and companies dealing with IT have been, more than ever before, gathering resources in order to deeply focus in this field.

Image visualization is a key component in self-driven car development. [11]. Therefore, it became obvious that if we are willing to hasten the process of having autonomous cars as safe as those driven by human beings, it is necessary to create precise object recognition models that would detect accurately and in real-time obstacles such as motors, bicycles, cars, trucks as well as pedestrians.

It is worth noting that recognition has been very challenging for a long span of time. This is argued to be grounded on the fact that objects in the images are not ordered like specific figures because in the real world we do not have regular designs. Many times,

the contents of an image do not keep the same forms when observed from different angles. Objects also appear to take different forms in terms of shape when exposed in conditions that vary in lighting, climate (rain, snow etc.) and when there is the phenomenon of occlusion. For this reason, recognition of objects is a very challenging task.

A study conducted by Isik et al. [8] remarked that humans tend to construe visual information very fast, effectively and in a subconscious way. Some contemporary research in the discipline of neurophysiological has also argued that visually, we are able to distinguish objects within 1/20th of a second.

The human beings' visual perception accommodates the ability to conduct difficult tasks such as driving using very little effort. A lot of work and effort has been put to advance the area of deep learning as an alternative and competing mean to humans' capabilities when it comes to recognition within the human environment. Till now, pioneer systems dealing with most of the issues related to visual perception have been led by models based on convolutional neural network systems, commonly known as CNN. The key strength of using CNN in automated object recognition, is the vigorous performance combined also with high detection speed and moderate memory requirements.

1.2 Research objectives and questions

The overall objective of this research is to explore and propose deep learning models that can be used to improve real-time obstacle recognition in driving field (e.g. recognition of cars, bicycles, pedestrians etc.). This goal will be approached through reviewing the studies that have been conducted on deep learning and object detection, perform experiments on specific models and provide results.

The below research questions have been developed based on the insight that this study will address the challenge of obstacle recognition for autonomous driving, with the ultimate aim of proposing systems with detection speed of 25 frames per second and above, as this framerate is a key element in deciding on the object recognition model.

The main objectives are:

a) Explore the available real-time speed object detection models
b) Train the state-of-the-art models on a representative driving dataset
c) Test the models and evaluate their performance.

d) Investigate in detail the results, perform a comprehensive analysis and extract valuable conclusions

1.3 Literature Review

In this section, we will present the evolution of image recognition in recent years and the most considerable attempts in object detection will be assessed. Object detection algorithms are broadly split into two categories: ”traditional” (Non-CNN based) and CNN based models.

1.3.1 The Evolution

Over the last decades, image recognition research field introduced several advances. In 1980, a pattern recognition model known as Neocognitron was created by Kunihiko Fukushima [3]. It posed abilities to identify particular contents, even in instances where the patterns shifted their positions and the shapes were distorted. Core parts of Neocognitron were the input layer and a sequence of connected layers where the training procedure implemented using an unsupervised technique. Biologically inspired (as animals' visual processing was great motivation for this model) the network's layers included two types of cells: simple and complex ones. The simple cells were extracting local features of the input while the complex ones were handling the combination and the translation of the different features [3]. An illustration of the model is shown in Figure 1.1. Five years later, in 1985, a supervised learning system was proposed by Rumelhart et al. taking advantage of the gradient descent method [17]. This algorithm was later used by LeCun et al. in a multilayer neural system designed to carry out tasks in recognition of handwritten digits [9]. The system however experienced major issues associated with its performance. The poor invariance regarding deformations or translations or of the inputs was a critical factor in terms of handling a variety of handwriting samples. The model required fixed-sized input images and they had to be centered in the input field. In 1998, LeCun et al. realized that in many object recognition problems the main focal points are the local features and the deformable organization of them, so the network's design should take this into consideration.

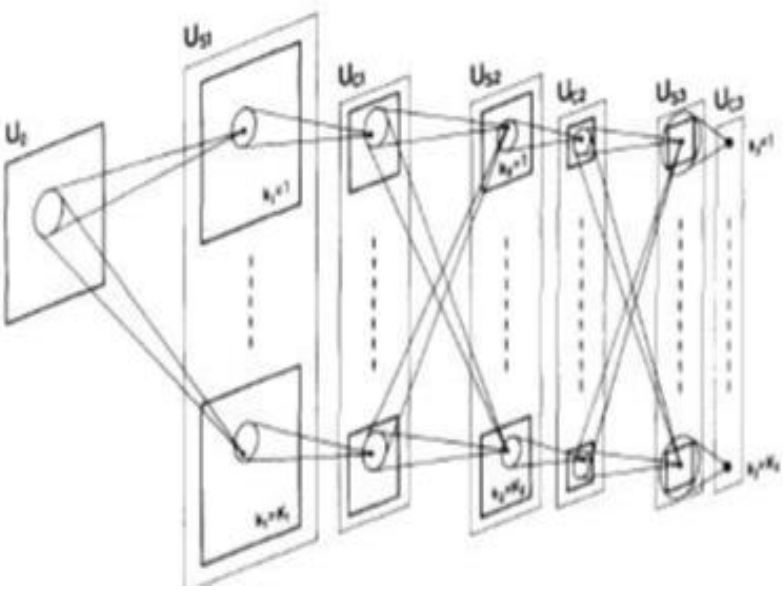


Figure 1.1: Fukushima's Neocognitron model [3]

As a result, they proposed the LeNet (Figure 1.2) that led to the "birth" of the well-known Convolutional Neural Network (CNN) [10]. Unfortunately, CNNs fell out of fashion soon and their usage was largely abandoned, as their potential was hidden by the lack of training examples and weak hardware of that period.

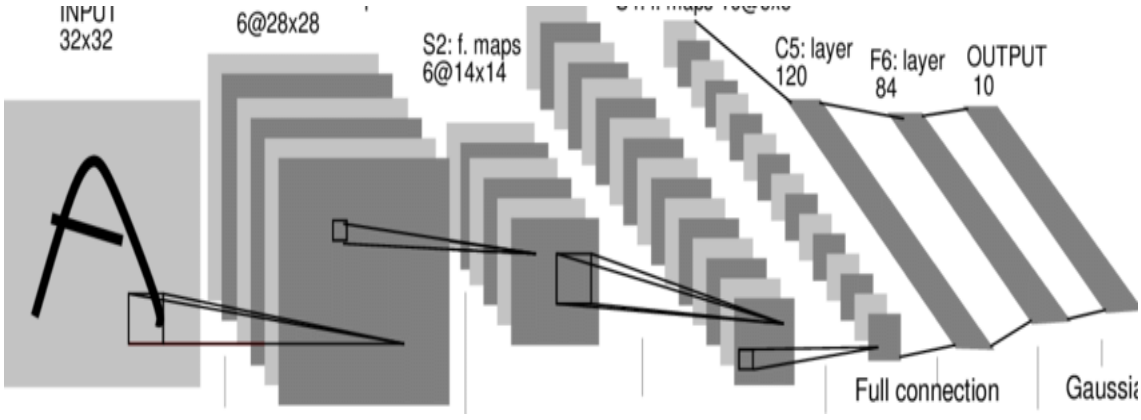


Figure 1.2 LeNet Architecture [10]

Prior to the year 2012, the object recognition techniques were following the feature-extraction-plus-classifier standard. Feature extraction task was handled by descriptors such as scale-invariant feature transform (SIFT) [22] and histograms of oriented gradi-

ents (HOG) [23] whereas classification task was performed usually by Support Vector Machines (SVM). These techniques, however, lacked robustness in relation to occlusion and deformation, and their performance could not generalize to a large set of classes.

In the year of 2012, new hope for CNN's potential was awoken when Alex Krizhevsky et al. won the ImageNet Classification Challenge, using a CNN based model which outperformed all the other competitors by a large margin [5]. In Large Scale Visual Recognition Challenge 2012 (ILSVRC, 2012), a part of ImageNet was used for classification, which included 1.2 million images in one thousand categories [1]. Besides the great amount of training images, GPUs power had increased even more. Krizhevsky's model revealed CNN's great capability in images classification when sufficient training samples are in place along with powerful GPUs utilization.

This fact marked the beginning of an era where CNN algorithms became the major tool used in image classification. As from year 2012, the focus of the research community on CNNs led to the proposal of exceptional models such as ResNet [18], Overfeat-Net [15] and VGG-Net [16] with remarkable results on single object detection.

Next challenge for the researchers was now the multiple objects detection. Regions proposal was the main technique implemented in these experiments in order to demonstrate candidate regions where classification task would be applied on, in a later step [6]. These region-proposals-classification approaches were capable to achieve high precision [16], however the time-consuming region proposal part slowed down the whole system, limiting them from applying to tasks where detection speed is critical, such as surveillance systems, autonomous-driving etc.

1.4 Multiple Object Detection models

In this section we will analyze the most significant models that contributed in the research of multiple object recognition field.

1.4.1 Regions-based Convolutional Neural Network (R-CNN) [10]

In 2014, Girshick et al. from UC of Berkley published a model that structured in accordance to the regions' proposal plus classification paradigm. Their method consisted of an independent algorithm for regions proposal and a convolutional network for ex-

traction of fixed-length feature vectors. Each vector was then fed into a Support Vector Machine (SVM) to classify the presence of the object within a candidate region. Regarding training, ILSVRC 2012 was used for the pre-training of the CNN while for the fine-tuning of the model a smaller dataset (PASCAL) was used.

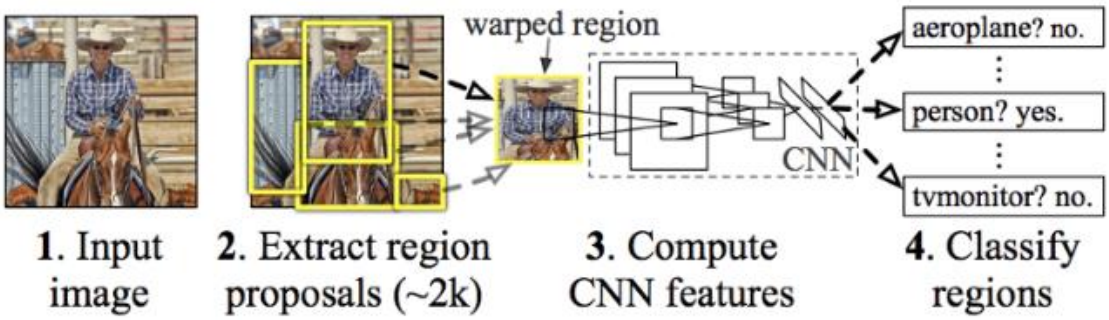


Figure 1.3 R-CNN architecture [6]

During test stage, Selective Search [19] was being applied for generating 2000 fixed size regions, which would possibly contain objects. Afterwards, all the potential regions were converted into feature vectors by the CNN extractor and classified by the SVM. In the last stage, the classified proposals were reduced by applying the greedy non-max suppression (NMS) algorithm, sorting the results by confidence scores, eliminating duplicate detections and removing the ones with an IOU (Intersection-Over-Union) lower than a specific threshold (Figure 1.3).

Compared to other models of that period, R-CNN was able for excellent detection accuracy. However, the model had some major drawbacks and one of them was the high computational cost. The need for an external method for regions prediction, burdened the whole model during training and testing time. Moreover, individually training of the Convolutional layers and the classifier along with the creation of approximately 2000 regions during test time was resulting to a very slow detection system.

1.4.2 Spatial Pyramid Pooling (SPP-Net) [7]

In 2015, He at al. proposed a new algorithm surpassing the fixed-size input limitation the models of that time were facing. Generally, the convolutional layers of the models can process images regardless of their size. However, this rule does not apply for the Fully Connected (FC) layers or the SVM, where input size should be standard. This

problem was initially solved by implementing some common practices such as cropping or wrapping (these techniques can be observed in Figure 1.4), but vital information was removed or deformed in the processed samples. As a result, the performance of the network was significantly affected.

Figure 1.4 Left: Image cropping technique - Right: Image warping technique

In order to further solve the above challenge, the SPP-net introduced a new layer which replaced the sliding window pooling layer, named as spatial pyramid pooling layer (SPP). It can be considered as a version of Bag of Words (BoW) [20] where bins' number connected to the FC layers or the SVM is fixed but the network is able to process various sizes of input images. Figure 1.5 shows a comparison between the structure of a conventional CNN and the SPP-Net.

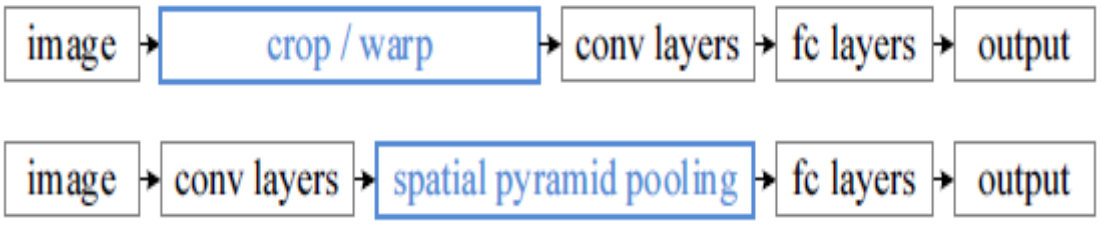


Figure 1.5 Structure comparison of a conventional CNN and SPP-Net [7]

Despite the development that SPP-Net has brought in CNNs, it faces the same drawbacks as R-CNN. Regional proposal is still performed by an external algorithm and also the Convolutional layers are trained separately from the classifier.

1.4.3 Fast Region-based Convolutional Network (Fast R-CNN) [5]

In 2015, the Fast Region-based Convolutional Network (Fast R-CNN), an innovative model considered as the development of R-CNN and SPP-Net, was introduced. The main idea was to pass the whole image to the CNN instead of processing each region proposal separately. The model maintained the fundamental notions of R-CNN and introduced many refinements. As in R-CNN, regions were proposed using an external algorithm. In Fast R-CNN a new pooling layer was used, named as Region of Interest (ROI) pooling layer. Single scale pooling with ROI layer allowed the loss error to be propagated in order to update the Convolutional layers during training. As a result, the end to end training became possible. System's architecture is presented in Figure 1.6.

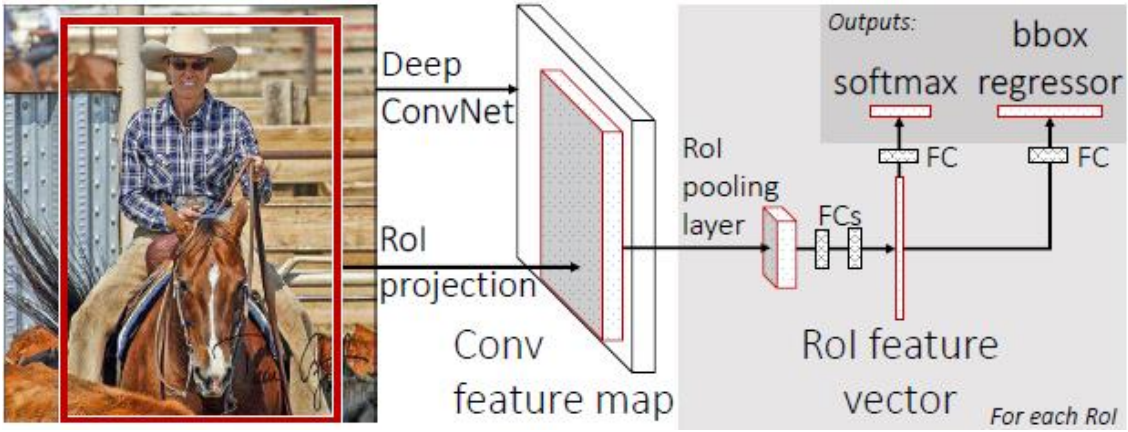


Figure 1.6 Fast R-CNN model architecture

1.4.4 Faster Region-based Convolutional Network Region-based Convolutional Network (Faster-RCNN) [14]

Faster-RCNN, created by Ren et al., revealed an alternative solution for the external region proposals method used in all former models, which was a bottleneck in terms of processing time. The concept was to utilize shared convolutional layers for region proposal as well as for detection. As in Fast R-CNN, the feature extractor output was exploited in order to perform classification/regression tasks but in this model it was also employed for the region proposals procedure. The authors named this shared convolutional network as region proposal network (RPN). A visual representation of RPN can be found in Figure 1.7. Firstly, the features were fed to a 3 × 3 convolution layer, fol-

lowed by a 1 × 1 convolution layer, with the output providing k bounding boxes and object existence scores for each region [53]. The hyperparameter k, defines the number of anchors. For the detection part, the same architecture as in Fast R-CNN was applied (Fast R-CNN subnetwork). All the high confidence scored proposals were forwarded to the last layers for classification and further bounding box refinement.

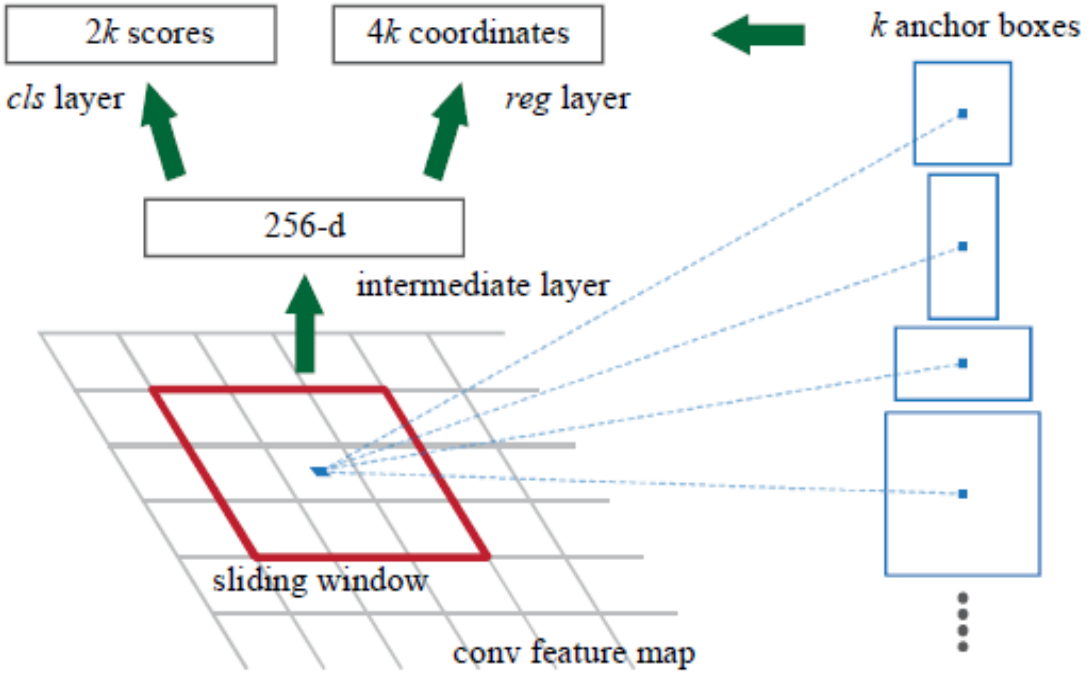


Figure 1.7 Regions Proposal Network (RPN) [14]

The model achieved top detection results in terms of accuracy. Furthermore, the inference speed was reaching the 10 frames per second, which made Faster R-CNN the fastest detector available in 2015.

1.4.5 Region-based Fully Convolutional Network (R-FCN) [21]

In 2016, Region-based Fully Convolutional Network (R-FCN) was introduced to improve Faster R-CNN's test speed due to the fact the relatively slow speed was limiting its usage in many sectors. Similarly to the Faster R-CNN, a regions proposal network was attached to the output of the feature extractor in order to generate region proposals. In R-FCN though, the Fast R-CNN subnetwork was now removed, and its place took a fully convolutional network in order to extract "position sensitive" feature maps as well

as to perform classification. We should mention that in R-FCN the final pooling layer was the only per-region process, a considerable factor that resulted to important reduction of the computations [53]. Subsequently, training procedure was faster and detection speed was increasing.

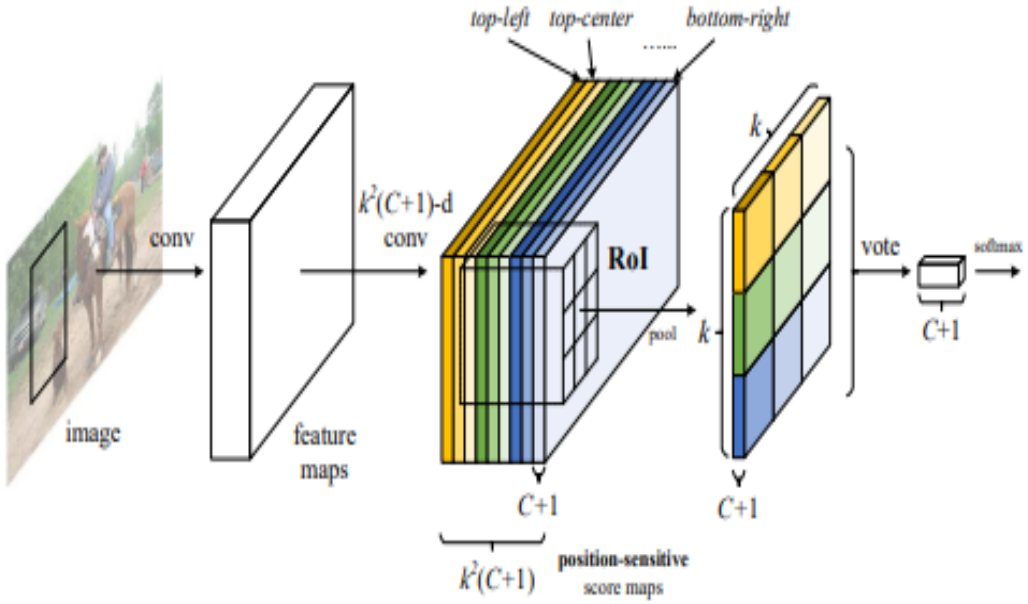


Figure 1.8 R-FCN model Architecture [21]

1.4.6 You Only Look Once (YOLO) [12]

In 2016, Joseph et al. proposed a new detector by reframing detection problem as regression problem. This innovative model was 10 times faster than the other state of the art networks. YOLO processes the input image and provides directly the class probabilities and locations. To achieve this, the input image is initially divided into a S x S grid and then processed by a 24 convolutional layers system followed by 2 fully-connected layers. An illustration of this process is shown in Figure 1.9. The output of the fully connected layer is a tensor with height S, width S and depth of (Bx5 + C). For each cell of this grid, B bounding boxes are predicted along with each box's object containing probability. The character C represents the number of classes and the digit 5 (4+1) relates to the 4 coordinates of each box plus the aforementioned probability score. In YOLO, there is no need for a class agnostic regional proposal network. It creates probability distributions and provides harder offsets.

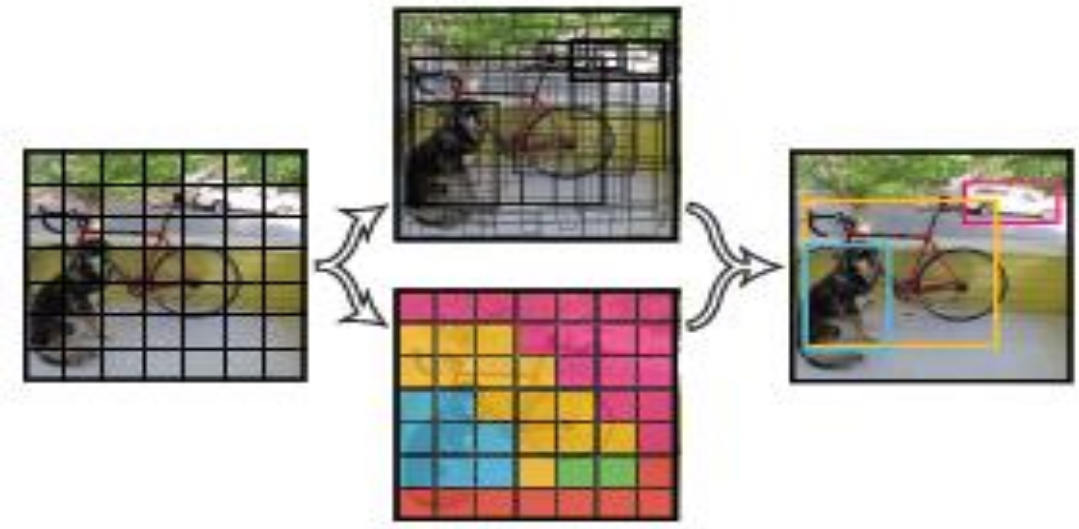


Figure 1.9 YOLO detection procedure: from the initial grid division to the final bounding boxes

During testing stage, it calculates class specific scores for each bounding box and returns the predictions with a score greater than a specific threshold. Experiments showed that YOLO has a high-speed detection capability, but its accuracy is not acceptable in many applications. A detailed representation of YOLO's architecture is depicted in below figure.

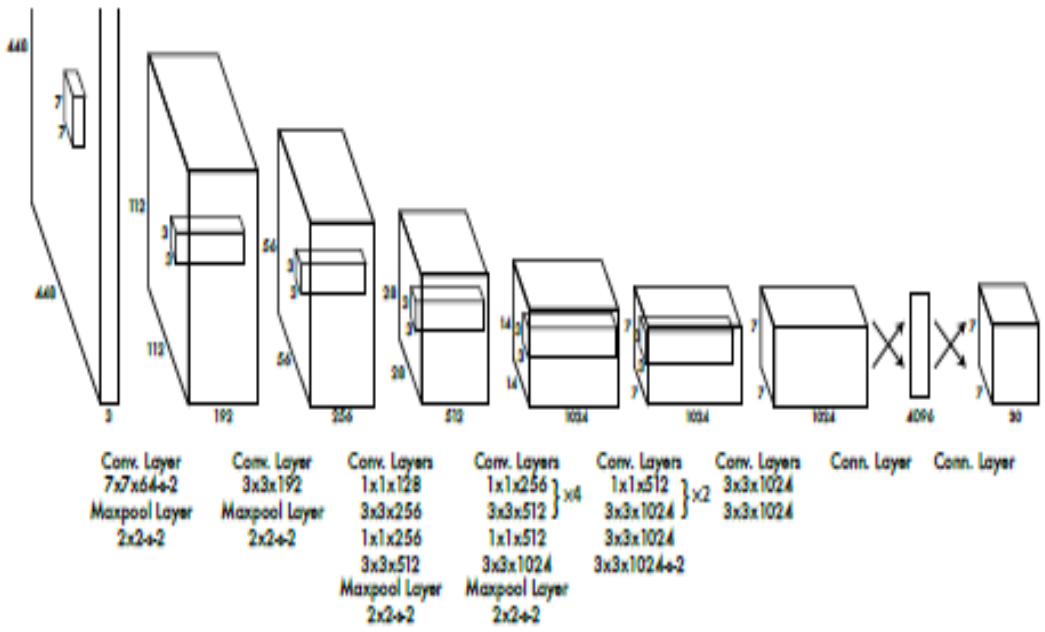


Figure 1.10 YOLO model architecture [12]

2 Deep Learning Background

In this section, we will introduce the basic concepts of Deep Learning (DL) and we will describe the basic functions of Artificial Neural Networks (ANN). Their role in Deep Learning will be assessed with particular focus on Convolutional Neural Networks (CNN), which are widely used in object recognition tasks.

Learning algorithms are broadly used in computer vision. In order to better understand the concepts and applications of deep learning though, it is important to have a brief review on its parent field of study, Machine Learning (ML).

2.1 Machine Learning Analysis

Nowadays, as data availability and computational power has increased, machine learning becomes more and more ubiquitous in information technology. Machine Learning is a scientific approach providing solutions to problems using machines. By machines we mean computer algorithms that systems use to progressively increase their performance on a specific challenge. It can be considered as a practical tool for modelling problems that are otherwise difficult (or sometimes impossible) to formulate. The algorithm builds a mathematical model from the available sample data, given for the specific task, known as "training data". Afterwards, it is able to provide predictions and/or make decisions without explicitly being programmed to perform on this task, as occurs in classical programming. Tom Mitchell in his book, provides a definition where a machine learns from experience ‘E’ regarding a given task ‘T’ and measured performance ‘P’, if its performance at the given task ‘T’, improves with a progressing experience ‘E’ [24]. There are two main learning techniques applied on machines in order to be able to handle tasks, the supervised and the unsupervised learning.

2.1.1 Supervised Learning

In supervised learning, the main objective is to find a function that maps the inputs to the outputs, based on training data [50]. The training data are usually consisted of an input (i.e. image, a vector etc.) and a label which defines the desired output (class) [25]. The algorithm analyzes the available data and returns a model which can be used for

testing new samples and predicting the output. In case the algorithm is able to generalize from the training samples to brand new examples in a reasonable way, the model will correctly determine the output. The most significant tasks performed in supervised learning algorithms are classification and regression [28]. In classification, the system predicts the class of a new previously unseen input based on its "experience" on the training data. In regression, the algorithm predicts a continuous output instead of discrete classes.

2.1.2 Unsupervised Learning

In this technique, we provide samples to the system, or as rightly stated in literature a population, but there are no corresponding labels available. The main goal in unsupervised learning is to model the distribution and the structure of the data in order to observe possible valuable properties. The algorithm groups the unsorted information taking into consideration differences and similarities of the data and returns a result pattern or clusters [28]. In this technique, no model for testing is produced, as it was in supervised learning. In case we want to add a new test sample, the algorithm should run again along with the already known patterns/clusters. Unsupervised learning is a very good approach for graphs and dimensionality reduction.

2.2 Deep Learning Analysis

Deep learning is part of the wide machine learning family, based on a set of algorithms that learn data representations by applying supervised, unsupervised or semi-supervised learning [26]. These algorithms are inspired by biological nervous systems, thus main architectures consist of multiple layer neural networks.

2.2.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) constitute the fundamental principle of Deep Learning. Neurons, which are the primary inter-connected computational units of ANNs, have multiple inputs and a single output (Figure 2.1), and typically they are distributed into layers. The information, which enters the system through the input layer, is processed by multiple intermediate - hidden - layers, ending up to a final layer which returns the mapping output values of the input information's extracted features. The processing stage includes transformations using the neurons' trainable parameters: weights and biases. Counter to hyperparameters, trainable ones are updated during

training cycles in order to achieve optimal predictions related to the input data. This is accomplished by quantifying the difference between targets and outputs given by the ANN, through a differentiable loss function.

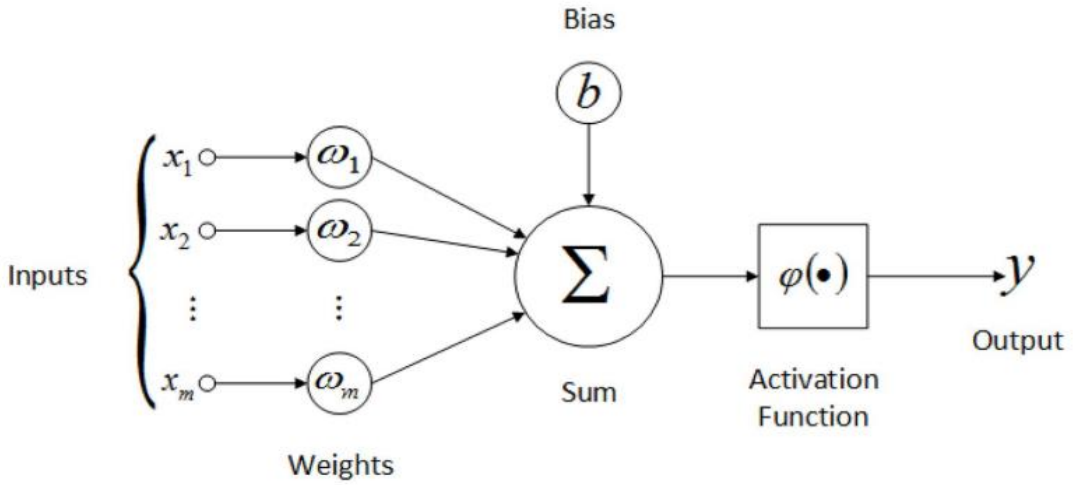


Figure 2.1 Mathematical model of artificial neuron [27]

In simpler words, the network is trained by updating its parameters, using the errors between ground truths and predictions. In an effective training, the error will gradually reduce until a certain point where there would be no obvious change. In that stage, the training process will be completed.

Neurons in generic neural networks are fully connected. This means that between any two consecutive layers, every pair of neurons is connected. A visual representation of fully connected neurons is shown in Figure 2.2. For instance, two adjacent layers consisted of i and j number of neurons respectively, will have i x j total number of connections.

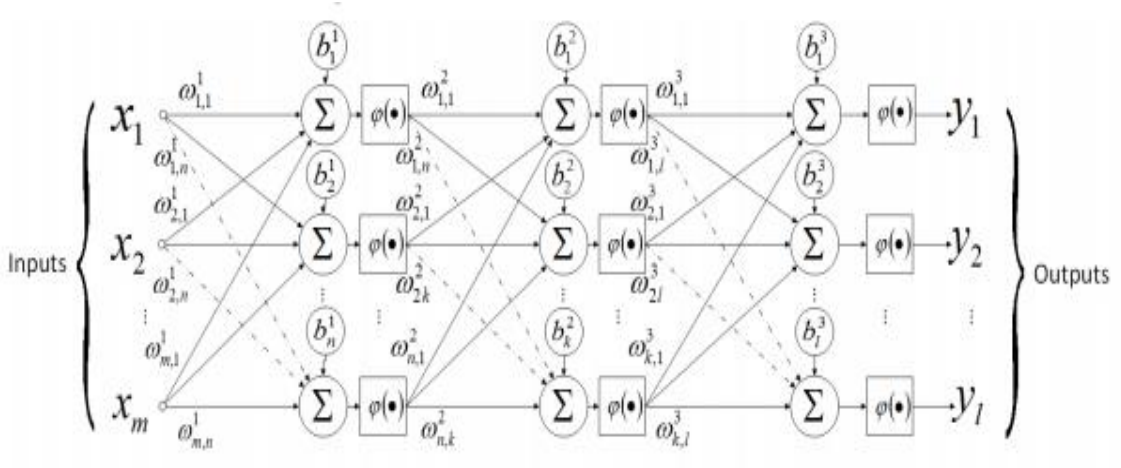


Figure 2.2 Generic multilayer feedforward neural network [41]

The outputs from the former layers become the inputs of latter layers. When information is “inserted” in first layer, simple computations are made, and the information passes to the second layer where more complex computations are conducted, based on the previous simple ones though. As a result, through each layer the network can make computations with increasing complexity in order to provide the best output. In this way, as it is also scientifically proven, the deep neural networks are able to propose solutions in various sophisticated tasks (natural language processing, image classification, recommendation systems etc.).

2.3 Convolutional Neural Networks

Convolutional neural networks (CNN) were proposed as a method able to transfer the robustness of artificial networks into computer vision applications, without massive inflation in parameter’s number and detection time. The idea behind CNN structure was conceived from the biology concept of the receptive field, a feature of animals’ visual cortex [29]. Receptive fields are detectors sensitive to certain types of stimulation, such as edges. In computers, this function can be approximated using the mathematical convolution operation [30]. The basic operations and layers of a CNN, briefly displayed in Figure 2.3, are described in the following subsections.

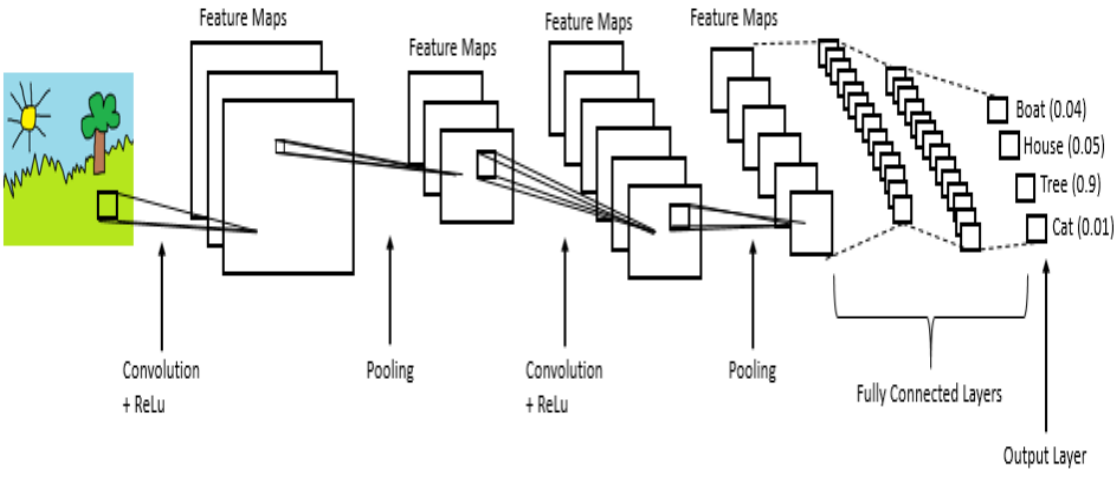


Figure 2.3 Architecture of a CNN [35]

2.3.1 Convolution and Convolutional Layers

Convolution is the core of a convolutional layer. There are three key elements in this operation: the input image, the feature detector or filter and the output, usually referred to as the feature map or activation map. In computer vision tasks, the input is a two-dimensional array of pixel values in case of grayscale image or a three-dimensional array in case of RGB-image, where the third dimension refers to the color channel. The filter is also an array (usually square sized) of trainable parameters, generally called weights, employed to extract features of importance for the current task. As the filter slides across the width and height of the input image from the top left corner down to the bottom right one, it calculates the dot products of the weights and the corresponding pixel values, and once all array's size is filtered it returns the feature map, which actually represents the "responses" of the feature detector in every spatial position. The size of the extracted feature map depends on the input's dimensions as well as the stride applied to the convolution. By stride we define the number of pixels the filter shifts per slide. In Figure 2.4 we can see an example of convolution using a 3x3 filter and the stride value is set to 1.

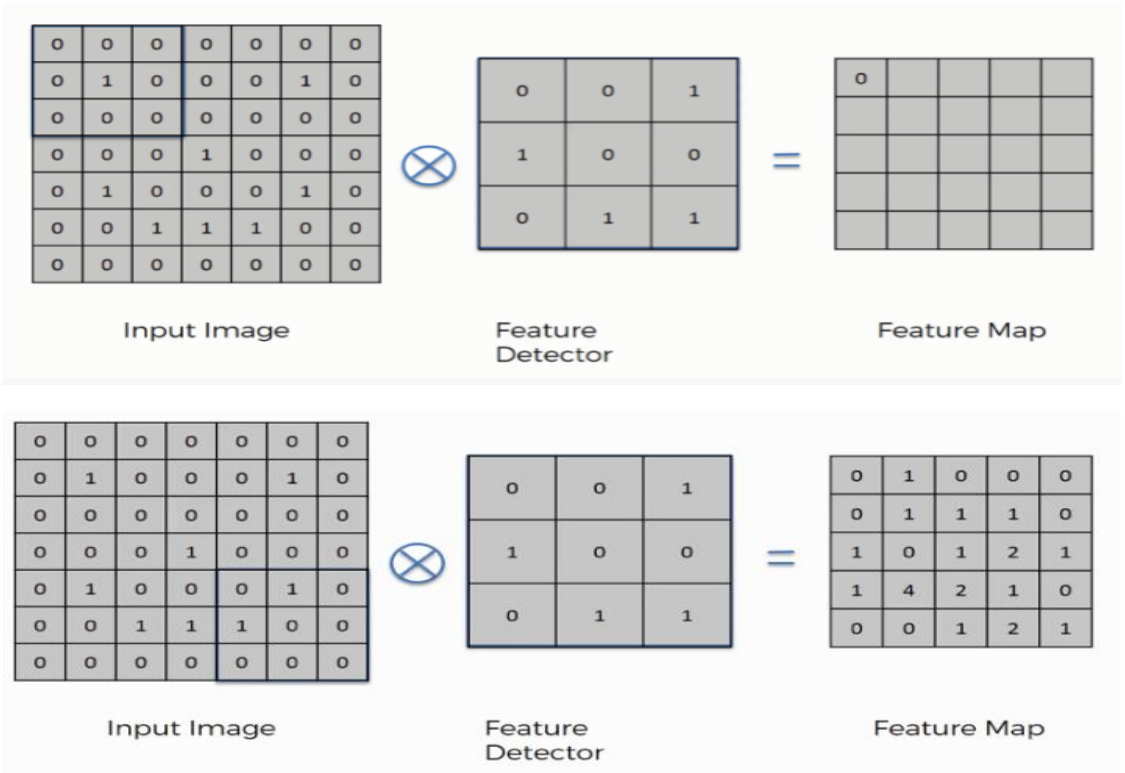


Figure 2.4. Convolution example using 3x3 filter and stride value 1 [31]

During training stage, the role of the filter is to teach the network to recognize a pattern, while during testing state, the filter searches if a specific pattern exists.

2.3.2 Pooling Layer

Typically, the feature map extracted after the convolution is inadvertently huge and the representations are not manageable to handle. In order to produce a denser but smaller and more useful version of the initial feature map, we apply the pooling method (down-sampling). There are two main popular pooling approaches, the max pooling and the average-pooling [32]. In max pooling, the input is divided into non-overlapping regions and the maximum value of each region is selected, while in average pooling the average value is calculated. We should mention that pooling doesn't have always effective results, as we can figure out from below examples (Figure 2.5 and 2.6). In both cases, we miss information by down-sampling the map so the applied technique we choose, can be critical.

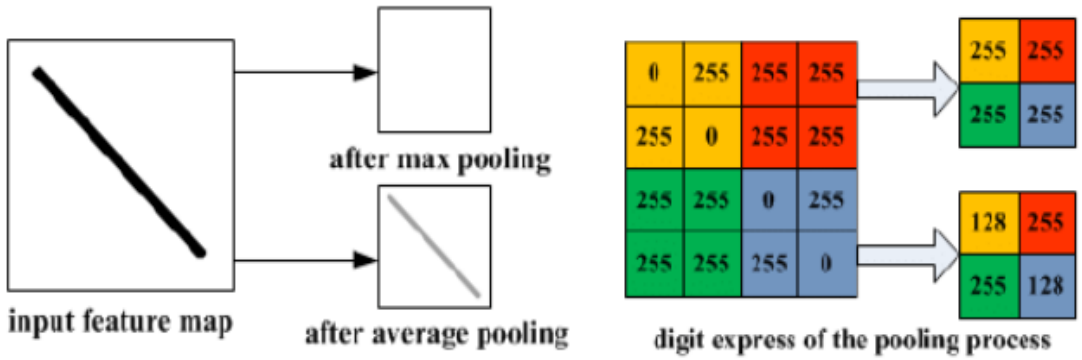


Figure 2.5 Drawback of max pooling [34]

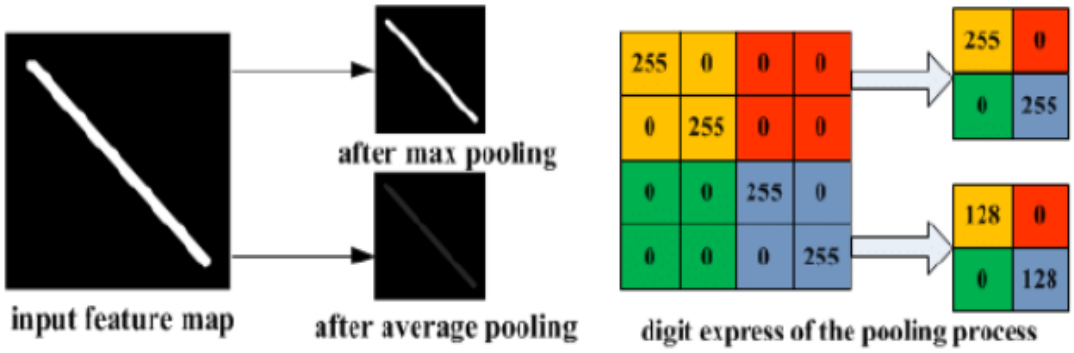


Figure 2.6 Drawback of average pooling [34]

2.3.3 Activation

Neural networks, including convolutional ones, rely on non-linear activation functions to obtain unique identification of features generated by each hidden layer. The input of these functions is normally the output of a convolution layer. Some popular functions used in CNN systems are: Sigmoid, Hyperbolic Tangent and Rectified Linear Unit (ReLU).

Sigmoid [33]

For a great period in neural network's history, sigmoid was the most broadly used activation function. Main reason is the interval it covers, 0 to 1. Therefore, it is suitable for models where we must predict the probability as an output, since probability values range from 0 to 1. It can be calculated by following equation : 1/(e^(-z) + 1) (Figure 2.7), which means that the output is tending to 0 for large negative values and to 1 for large positive ones. However, it is hardly applied nowadays due to the vanishing gradient problem it shows up along with its non-zero centered behavior.

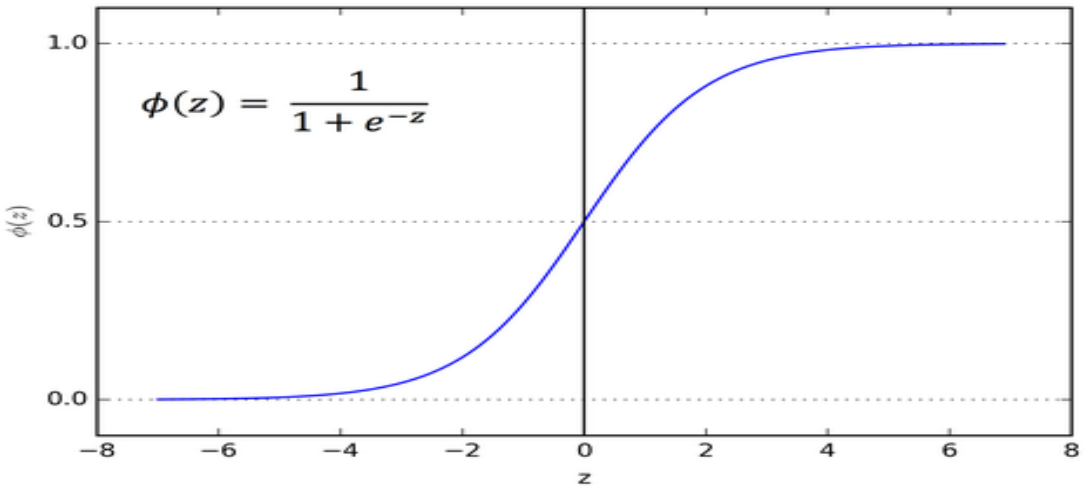


Figure 2.7 Sigmoid activation function graph

Hyperbolic Tangent [33]

We can consider hyperbolic tangent as a better version of sigmoid. It is defined as tanh(z) = (e^(2z) - 1)/(e^(2z) + 1), with outputs ranging in the interval [−1, 1] (Figure

2.8). Its main advantage over sigmoid is that the negative inputs are mapped as strongly negative ones and the zero inputs are mapped as near zero values. This function is mainly used in classification tasks.

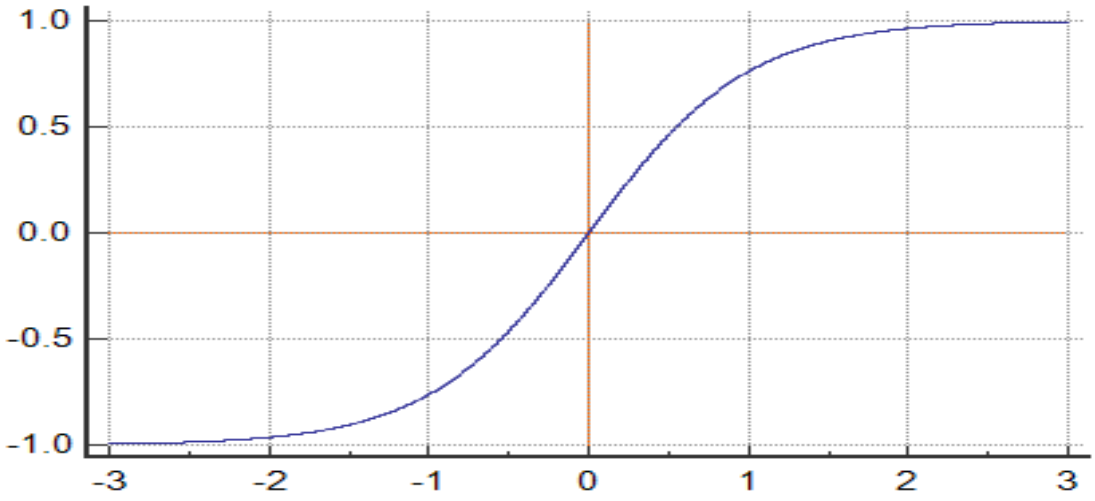


Figure 2.8 Hyperbolic Tangent activation function graph

Rectified Linear Unit (ReLU) [33]

ReLU is currently the most popular activation function as it is used in almost all the convolutional neural networks. It takes values in the range from zero to infinity and it is computed by the function f(z)= max (0, z). As we can observe in Figure 2.9, ReLU is half rectified. It means that f(z) maps to value zero when z is less than zero and takes value equal to z when z is greater or equal to zero. Here, the main issue is that all the negative values turn into zero, which reduces the ability of the model to train and fit to the data properly.

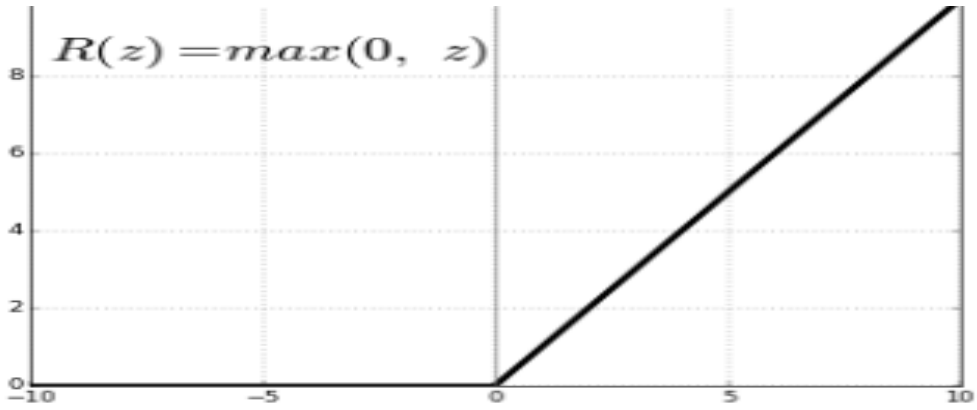


Figure 2.9 ReLU activation function

2.3.4 Fully Connected Layer (FC Layer)

This layer is usually employed at the CNNs as the output layer. Actually, FC layers are utilized in order to identify global configurations of the features discovered by the former layers of the network. It takes an input volume (depending on the output of the previous layer – convolutional, ReLU or pooling) and produces a N-dimensional vector, where N represents the number of classes that are the possible outcomes. For instance, in a digit classification application, the vector would have size 10 since there are 10 digits and each number in the 10-dimensional vector would provide the probability of a certain digit.

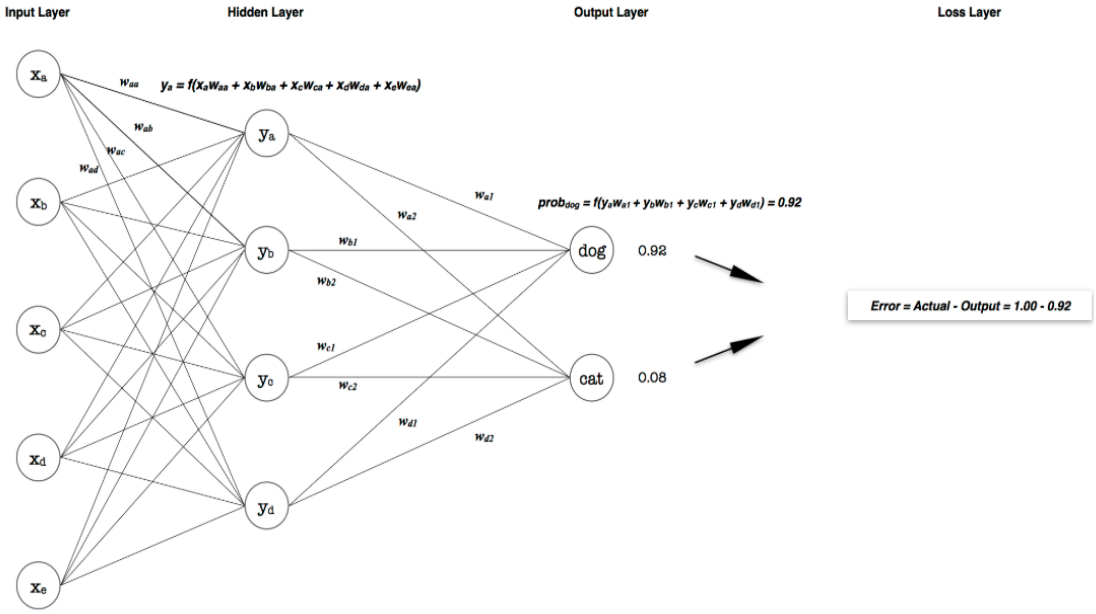


Figure 2.10 Fully Connected Layer diagram

Basically, a FC layer searches for the high-level features that most likely correlate to a class and by using specific weights it returns the correct probabilities for the different classes [36]. Figure 2.10 presents a toy example for better understanding FC layers function.

3 Models and Datasets

This chapter contains a theoretical analysis of the CNN-based models SqueezeDet, YOLO version 2 and YOLO version 3 which are practically implemented and tested in our experiments. Furthermore, it is provided a thorough description of the datasets ImageNet, Pascal VOC and especially KITTI which is used for the main training and evaluation part of the networks.

3.1 Models

The architecture of the three above mentioned networks chosen for our experiments is analyzed in this section

3.1.1 SqueezeDet [37]

SqueezeDet, which draws inspiration from YOLO, is a single-shot, fully convolutional, lightweight object detection pipeline, proposed by Bichen Wu et al. in 2016. As it belongs to the Single Shot Detector family, region proposal as well as classification tasks are performed by a single network. The core parts of SqueezeDet include: a convolutional layer attached to 10 adjacent fire modules and a final convolutional layer as output. In general, SqueezeDet is a small size detector due to the feature extractor it employs, which is based on SqueezeNet, constituted of sequential fire modules. Small is a relative term though, if we consider the fact it consists of approximately two million trainable parameters [38], nevertheless it is relatively small compared to other widely used models that reach the amount of 25 million or even 145 million parameters.

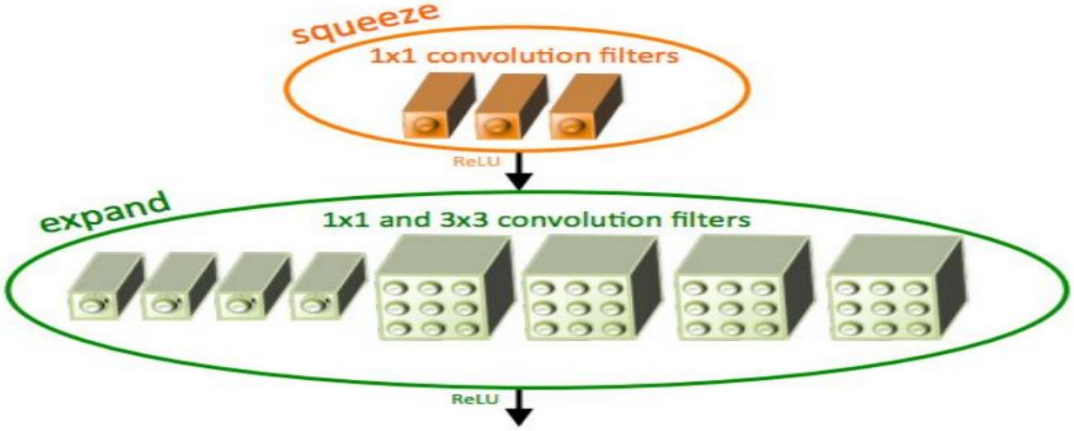


Figure 3.1 SqueezeNet fire module [39]

Fire Modules contain a squeeze layer as input, followed by two parallel layers, called expand layers (Figure 3.1). Input layer is a 1x1 convolutional layer and, as its name reveals, it squeezes the input tensor to a smaller volume one, without decreasing the spatial resolution. The expand layer, which combines two versions of convolution filters (1x1 and 3x3), receives the previously compressed tensor, extracts the important features and produces an activation tensor with large channel size.

A flowchart of SqueezeDet's architecture with the corresponding feature map dimensions is shown in Figure 3.2

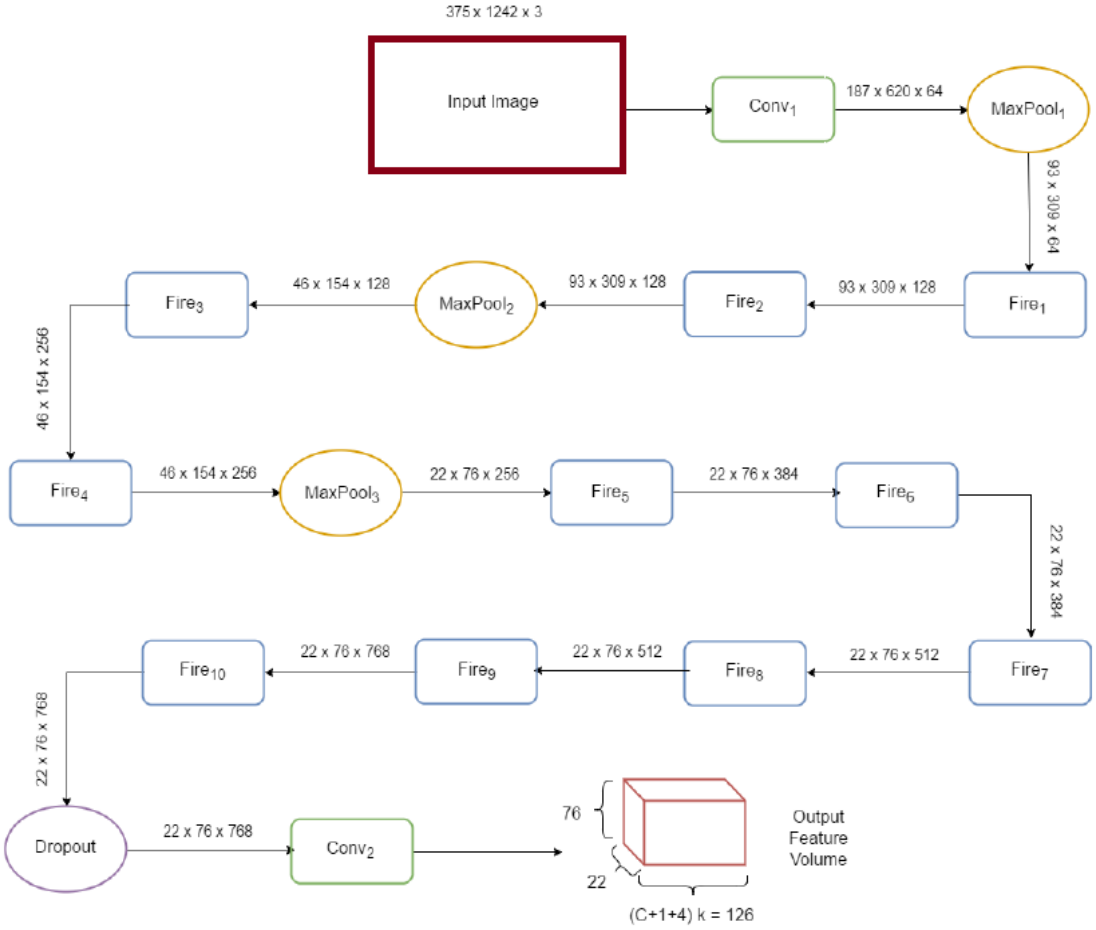


Figure 3.2: SqueezeDet Architecture [51]

SqueezeDet, along with all the other single shot detection systems, is an advantageous architecture compared to the traditional region proposal methods, due to the fact that the computations are completely shared between the regions of interest, resulting to lower detection time.

3.1.2 YOLO version 2 (YOLOv2) [40]

As we have seen in chapter 1.4.6 , Redmon, Divvala, Girshick and Farhadi define YOLO, as a system for object detection that facilitates real-time processing. This study unified the object detection components into a single neural network. The YOLOv2 framework is a modified version of original YOLO with the objective of improving its detection speed and accuracy.

A new element in YOLOv2 is the batch normalization practice [41]. Authors applied batch normalization after each convolutional layer of the first YOLO implementation in order to improve and regularize the model's convergence. Besides that, they also removed the dropout layer. This itself resulted in 2% mAP increase (compared to YOLO). Another key modification was the utilization of anchor boxes. In general, there are two main techniques for the bounding boxes prediction task: direct prediction of the object's box or using pre-defined bounding boxes, known as anchors. In YOLO the coordinates of detection boxes are generated through the fully connected layers on top of the convolutional feature extractor, a method resulting in serious amount of localization errors. In 2nd version of YOLO, these fully connected layers are removed, and anchors are used, providing better results. Another addition of this model is that the input dimension size varies from 320 x 320 to 608 x 608, images are randomly resized to that dimensions in order the network to be trained across a variety of input resolutions. This method increased the mAP by 1.5%.

YOLOv2 predicts the detections on a 13 x 13 output feature map. This size is enough for large objects but not smaller ones. For better localization of smaller objects, features from a former layer with size 26 x 26 is taken. This leads to a 1% performance improvement.

The main innovation in YOLOv2 is its "backbone", a new architecture named Darknet-19. Consisted of 19 convolutional layers and 5 max-pooling layers on top of the last convolutional layer (softmax layer for classification), it only requires 5.58 billion operations to process an image with great results [42]. A detailed view of the Darknet-19 structure and the full network can be observed in below figure.

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Figure 3.3 Architecture of Darknet-19 [40]

3.1.3 YOLO version 3 (YOLOv3) [13]

This is the third object detection model in YOLO family. The most noteworthy addition in this version resides in the introduction of the three scales detection system. Since there are objects of different sizes on the images, we want the model to detect them all, small and big ones. As the network goes deeper, its feature map gets smaller and it becomes harder and harder to recognize the smaller objects though. To support detection on varying object sizes YOLOv3, predicts boxes at 3 different scales (Figure 3.4). The features are extracted from each scale by using a method similar to that of feature pyramid structure [43] [54].

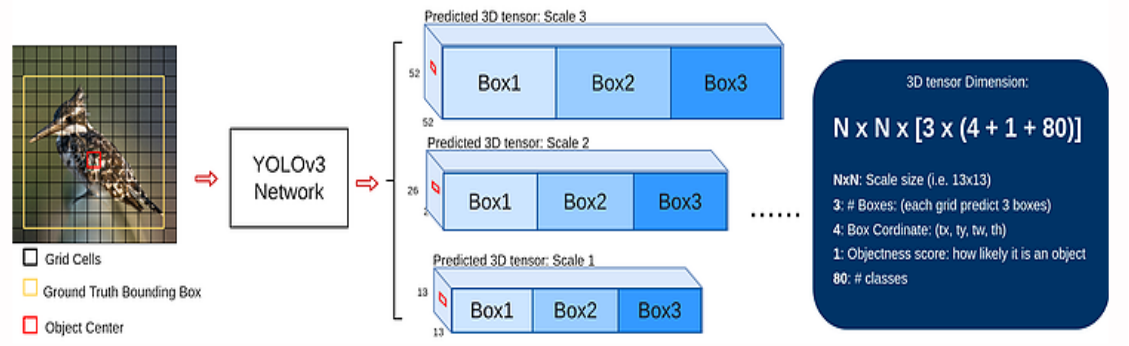


Figure 3.4 YOLOv3 three scale detection [51]

For feature extraction, it uses a Darknet variation consisted of 53 layers, named Darknet-53 (Figure 3.6.). The new key elements added on this network are the residual connections which they actually represent a shortcut path between layers. Without these, the network is a classic CNN which learns the feature one by another. As the network goes deeper, it becomes harder and harder to learn features well. If we add a shortcut, the layers in-between (as illustrated in Figure 3.5) will learn the “information” that should be added to the old feature in order to provide better ones [51]. This makes it easier for the network to learn the features stably, especially in very deep networks. Instead of learning a new complex feature, it learns the supplemental "residual" which is added to the old feature.

It is worth mentioning that in 3rd version of YOLO, softmax layer is replaced by 1x1 convolutional layer with logistic regression. When in a dataset we have similar classes, like person and woman, training with softmax might not let the network generalize the data features well. This led the authors to abandon softmax function and each class

score is now provided using logistic regression where a threshold is used to predict multiple labels for an object. Classes with scores greater than this threshold are picked and assigned to the box.

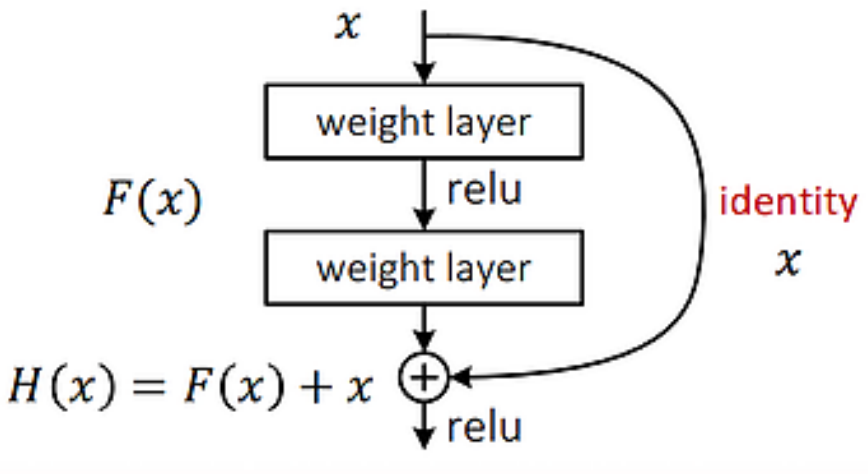


Figure 3.5 Residual Connection

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

,

Figure 3.6 Darknet-53 [13]

3.2 Datasets

In this section, the main datasets used during pre-training, training and testing stage of our experiment will be presented. When choosing a training/testing dataset for a specific application, it is of great importance to take into consideration the volume of the samples and their correlation with the application's target environment. In this project, we are dealing with the road obstacle recognition task in normal driving conditions. Consequently, we preferred the KITTI detection set over other available choices as our main dataset. Regarding pre-training, classification data-sets are widely used to initialize the parameters of the networks. Here, we consider the ImageNet 2012 challenge, MS-COCO and PASCAL VOC 07/12 data-sets for this objective.

3.2.1 KITTI Vision Benchmark Suite [49]

KITTI is a project of the Karlsruhe Institute of Technology and Toyota Technological Institute of Chicago. It is specialized in Autonomous Driving and was created to provide real-world benchmarks to the research community. By driving a car vehicle around the German city of Karlsruhe equipped with two high resolution grayscale and color cameras, they collected rich information containing images, optical flow, visual odometry data etc. Even though there is a little variation in terms of climate (rain, snow etc.) and lighting conditions, the dataset simulates a variety of driving environments. KITTI consists of eight different sets, each one having its own specific purpose. In this thesis, we used the 2D object detection benchmark. It includes 7481 training and 7518 test colored images in .png format, with each sample having size of approximately 800kb to 900kb and resolution 375×1242 pixels. Since there are not available annotations for the test set, in our experiment we split the training samples (7481 images) in training and validation set in order to be able to perform evaluation on the model. The labeled data include ground-truths for at most 15 cars and 30 pedestrians containing information for the class type, truncation, occlusion level, orientation and bounding box coordinates. The class type describes 8 possible types of objects: pedestrian, cyclist, car, tram, van, person sitting, truck and "Misc". The latter class defines objects that are not counted in evaluation metrics which may describe regions of distant objects or objects that don't correspond to the above-mentioned classes. In Figure 3.7 we can see some examples from KITTI 2D detection set.

Figure 3.7 KITTI dataset samples

3.2.2 ImageNet [1]

The ImageNet database contains over 15 million labeled high-resolution images of objects in roughly 22,000 categories. The annual Large-Scale Visual Recognition Challenge (ILSVRC) competition uses a subset of 1000 categories with 1.2M training images, 150,000 testing images and 50,000 validation images [55].

Even though it provides only single class prediction ground truths per image for the biggest part of its volume, it is widely used in training networks for computer vision tasks other than image classification, such as object detection and image segmentation, mainly on the pre-training procedure of the model. The importance of pre-training is described in Chapter 4.1.

In Figure 3.8 random samples of the database are presented.

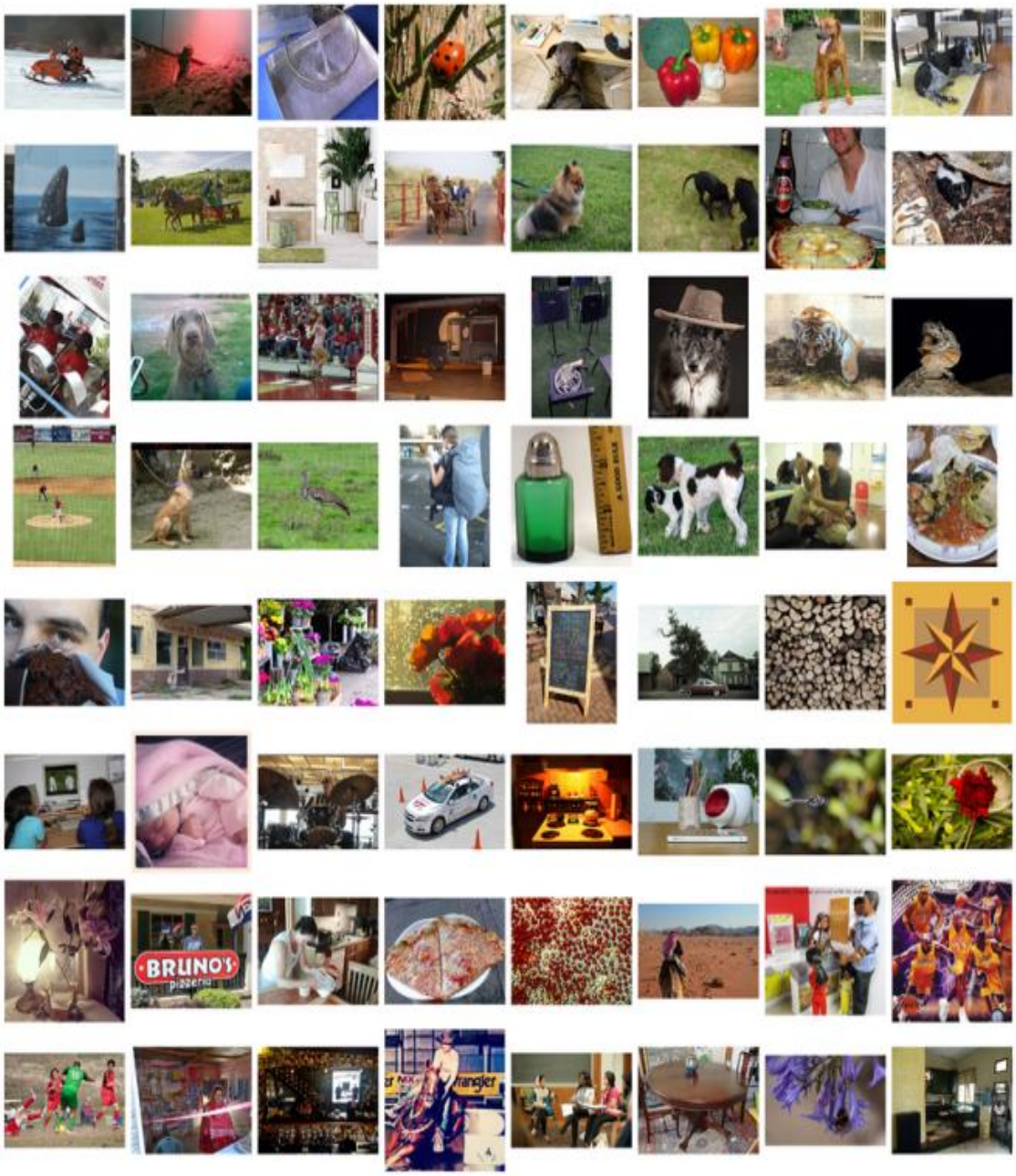


Figure 3.8 ImageNet samples [44]

3.2.3 Pascal VOC [45]

Pascal VOC 2007/2012 dataset is a significant foundation of Pascal VOC Challenge, which contributed in the development of image classification and object detection. In this thesis, we utilized the dataset from Pascal VOC 2007 and 2012 Challenges for pre-training purposes. They respectively include 9,963 (875MB) and 11,125 (1.9GB) images of 20 classes from 4 main categories: animal, person, vehicle and indoor.

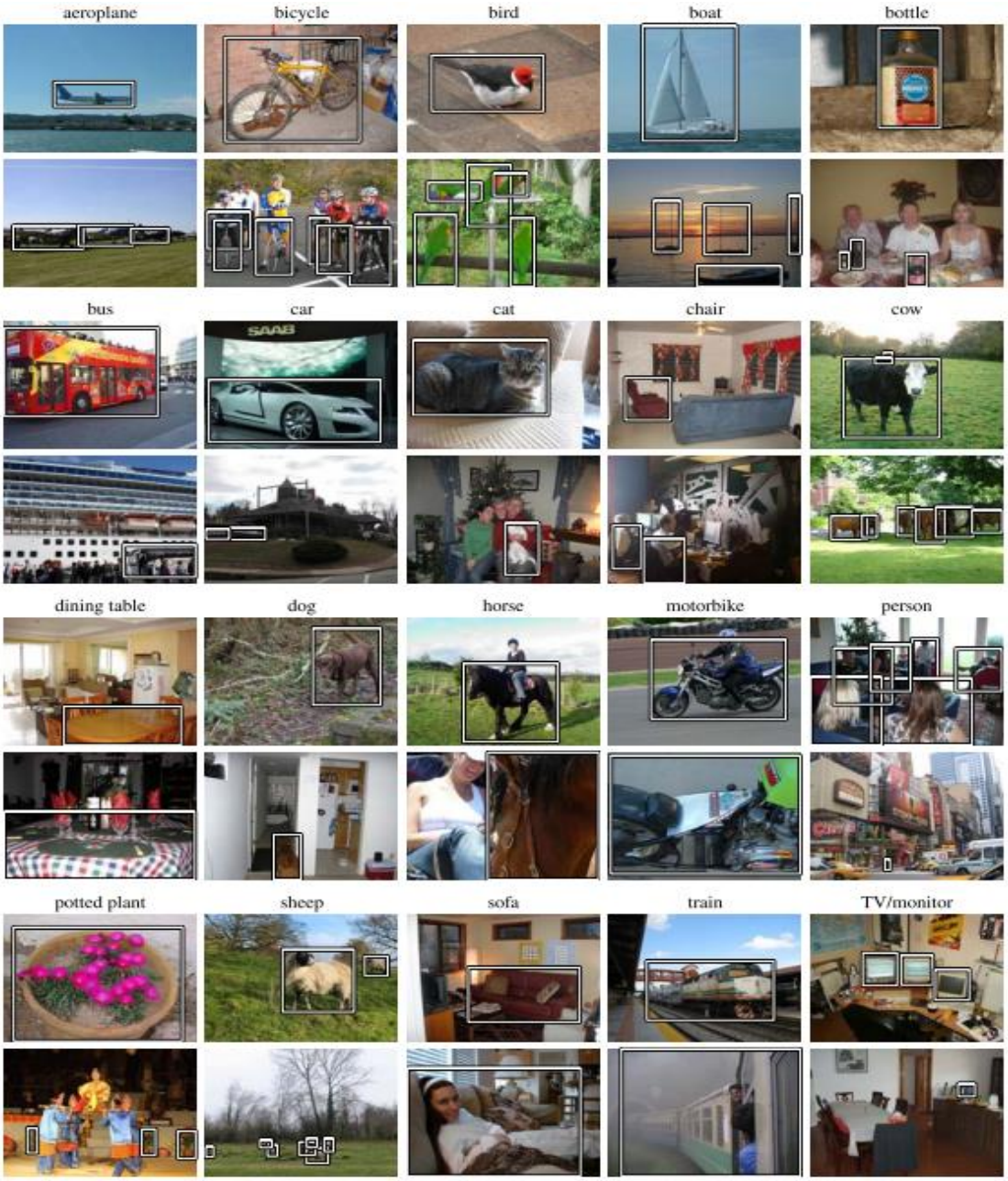


Figure 3.9 Pascal VOC 2007/2012 samples [46]

The training data consist of a set of labeled images where the annotation files provide the object's class and the bounding box coordinates. Figure 3.9 shows samples from Pascal VOC datasets for each class.

3.2.4 COCO [47]

COCO, which stands for Common Objects in Context, is a large-scale object detection, segmentation, and captioning dataset, created to boost the development of the state-of-the-art in object recognition. It includes images of complex everyday scenes enclosing common objects in their natural context. An illustration of these general content objects grouped in 80 categories are shown in Figure 3.10.

Figure 3.10 COCO labeled categories [48]

With a total of 2.5 million labeled instances in 328k images of general content, it is extensively utilized in novel interfaces for object detection. In this thesis, COCO is used as pre-training dataset for one of our experiments. In Figure 3.11 we can observe a representative group of COCO's images.

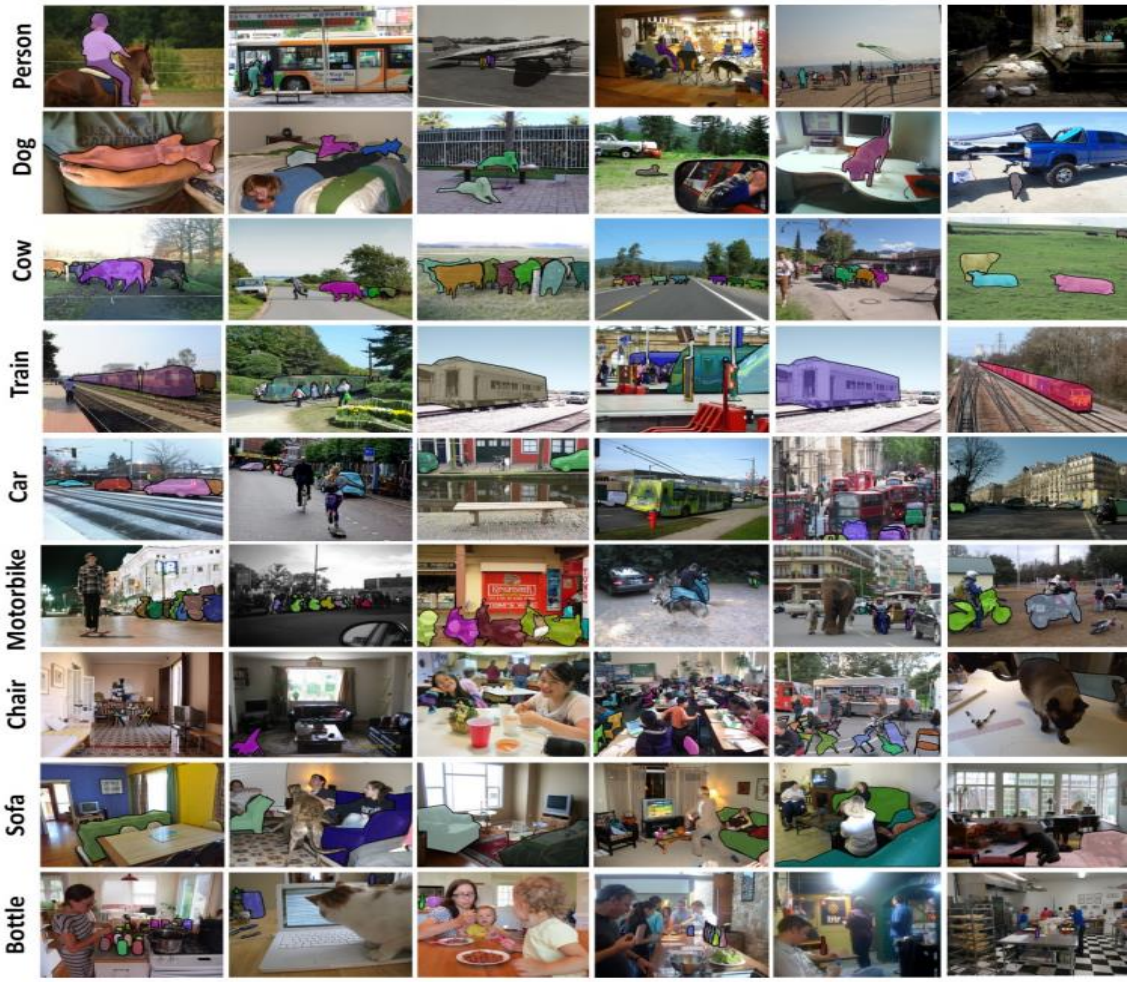


Figure 3.11 Pascal VOC 2007/2012 samples [47]

4 Experiments and Results

In this chapter, we will present the methods and the configurations followed during the training process of the models and we will analyze the testing results they achieved in our experiments. We have trained and tested three state-of-the-art object detection models in KITTI 2D object detection set in order to explore their capability in the driving field.

4.1 Hardware and Software Environment

Before we move on to the experiments' analysis part, we should mention the computer hardware our models were running on, plus the software tools used for the implementation of the networks.

Our initial hardware setup was a laptop computer with an Intel Core i7-7700HQ CPU running at 2.8 GHz, 8 GBs of RAM and a NVIDIA GeForce 1050 TI (CUDA capable) GPU with 4GBs VRAM. The operating system was Windows 10. However, the experiments we conducted were very demanding in terms of memory and computational power and our hardware (mainly the graphics card) turned out to be inadequate. Thus, a new setup was chosen, with 16 GBs of RAM and a NVIDIA TITAN XP (also CUDA capable) GPU with the remarkable 12GBs of VRAM which handled our experiments with much more ease than the first one. The operating system was Ubuntu 16.04.

All detection networks used in the experiments were implemented in Python on top of the TensorFlow library. TensorFlow is an open source software library released by Google Brain's team, focused on machine learning applications such as neural networks [52]

4.2 Pre-training

For the training process, two steps were followed: pretraining and main training. In general, when we want a new neural network to perform a task on a specific dataset, the training procedure starts from scratch using random weights and once it is completed, the weights (model checkpoint) are stored on the disk for future use. When we should proceed with another task on a different dataset, it is not convenient to repeat the train-

ing process from zero point. Instead, we can utilize the already pre-trained model's weights and optimize our net on the new dataset. This procedure is called transfer learning. Currently, the basic method to train a network on a new dataset is to pretrain the network on a set with great volume of samples and broad categories. Then, we initialize all the trainable parameters, except the ones from the model's last layers, using the already pre-trained weights. The last layers will be initialized with random weights. There are two benefits of pre-training a network in a big dataset.

Firstly, it is a great method to achieve good results on small dataset. The pre-training network can learn a variety of general, basic features (such as edges, arcs, circles and colors patterns) from a big dataset with wide variety of classes. Afterwards, we just perform the fine-tuning process.

Secondly, it can reduce training time. Even though GPUs these days are powerful, training a network from scratch could still last days to weeks. It is unproductive to wait for such a long period every time we amend some of the network's parameters.

4.3 SqueezeDet experiment

Our first experiment involved SqueezeDet network's training and evaluation on the three main classes of KITTI object detection set: Car, Cyclist and Pedestrian.

4.3.1 Training configuration

For the pre-training stage, we used the extraction model trained on ImageNet classification task. Regarding main training, we randomly split the images into 5237 (70%) and 2244 samples (30%) for train and test/evaluation procedure respectively (for detailed information about KITTI, please refer to Chapter 3.2.1).

In order to achieve the best possible result, we followed the literature and the well-known problem solving method "trial and error". After many attempts we concluded to the following configuration. We trained 200.000 batches, each batch consisted of 12 samples and the training stage had duration of approximately 55 hours. Initial learning rate was set to 0.01 with a weight decay of 0.0001 and momentum of 0.9.

The loss values and their trending line during the training period are shown on Figure 4.1.

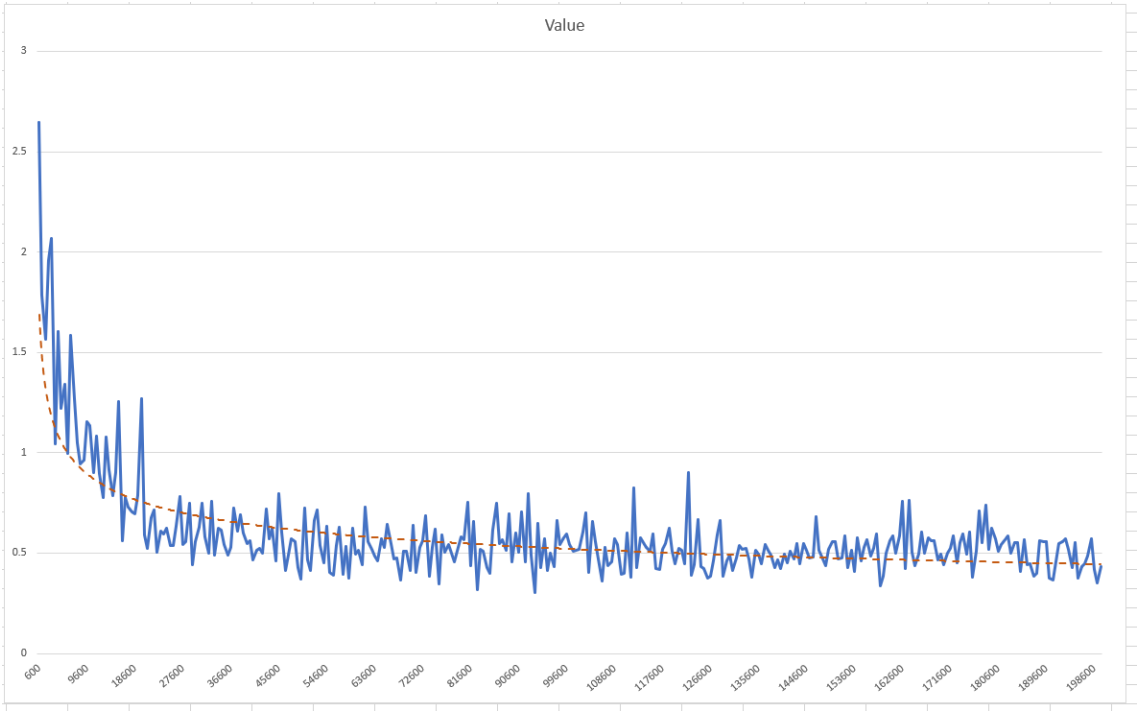


Figure 4.1 SqueezeDet's training loss values through the batches

4.3.2 Testing and Results of the model

Following the training, we run the model on the test set. SqueezeDet scored a mAP of 72.54%, with great precision in all three classes. Specifically, as we can also observe in Figure 4.2, the model located the position of the cars and identified their class with an average precision of 82%, the cyclists with 74% and the pedestrians with 62%. Moreover, the number of correct and false predictions for the 3 classes are presented in the same figure.

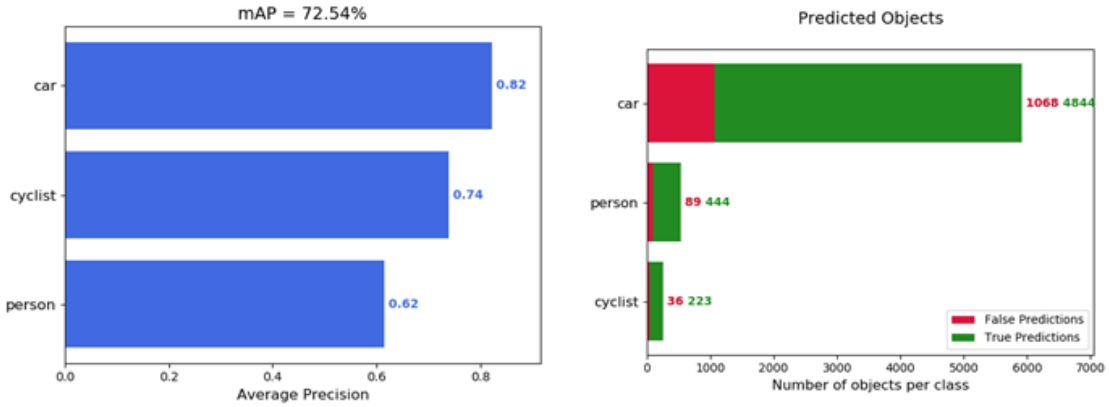


Figure 4.2 SqueezeDet evaluation results

In following figures, we are displaying some indicative results received during SqueezeDet's test stage and represent its detection capability on KITTI. The predicted bounding boxes are marked in green and the ground truth boxes in blue.

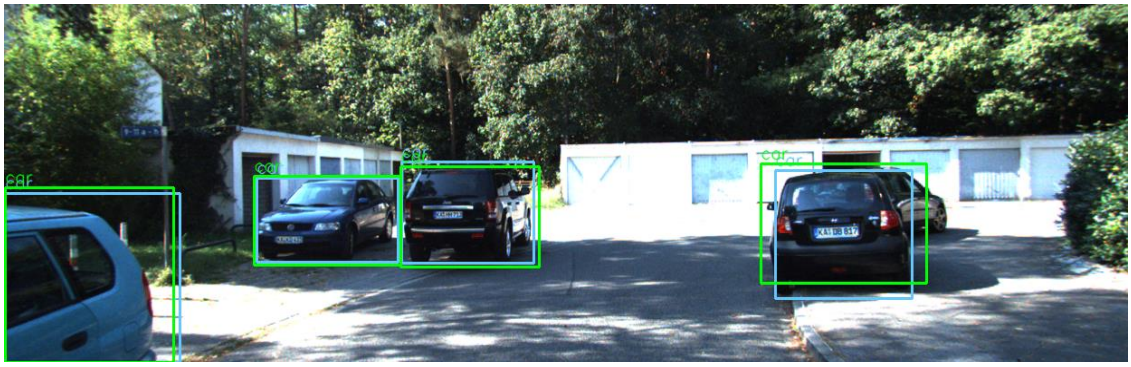


Figure 4.3. Result of Car class – The system locates the position of the cars and identifies the class correctly. 4 cars are detected (confidence threshold 0.5)

Figure 4.4. Result of Cyclist class – A true positive detection (single prediction evaluation procedure - confidence threshold 0.5).

Figure 4.5. Result of Pedestrian class – The model detects the class and the position of the two pedestrians accurately even one of them (left) is not fully visible (confidence threshold 0.5).

Next samples have been added to mark some cases where SqueezeDet failed to provide the desired results. Both detection and localization errors are observed.

Figure 4.6. Misclassification Result of Pedestrian class – The network wrongly recognizes the plant as a Pedestrian (confidence threshold 0.5).

Figure 4.7. Misclassification Result of Cyclist class – The sign is mistakenly identified as a Pedestrian (confidence threshold 0.5).

Figure 4.8. Result of Cyclist class – The predicted position of the cyclist does not match with the ground truth (confidence threshold 0.5).

4.4 YOLOv2 experiment

In this experiment, we explored YOLO's second version capability on KITTI using the same training and evaluation samples as previously.

4.4.1 Training configuration

Model's weights were initialized using pretrained weights on PASCAL VOC 2007/2012 (further information for this dataset can be found in subchapter 3.2.3). Total of 112.000 batches were trained with a batch size of 20 samples. This run took approximately 60 hours to finish. Learning rate's starting value was 0.001 with a weight decay of 0.0005 and momentum of 0.9. The training loss of this experiment is shown in Figure 4.9 along with the trendline.

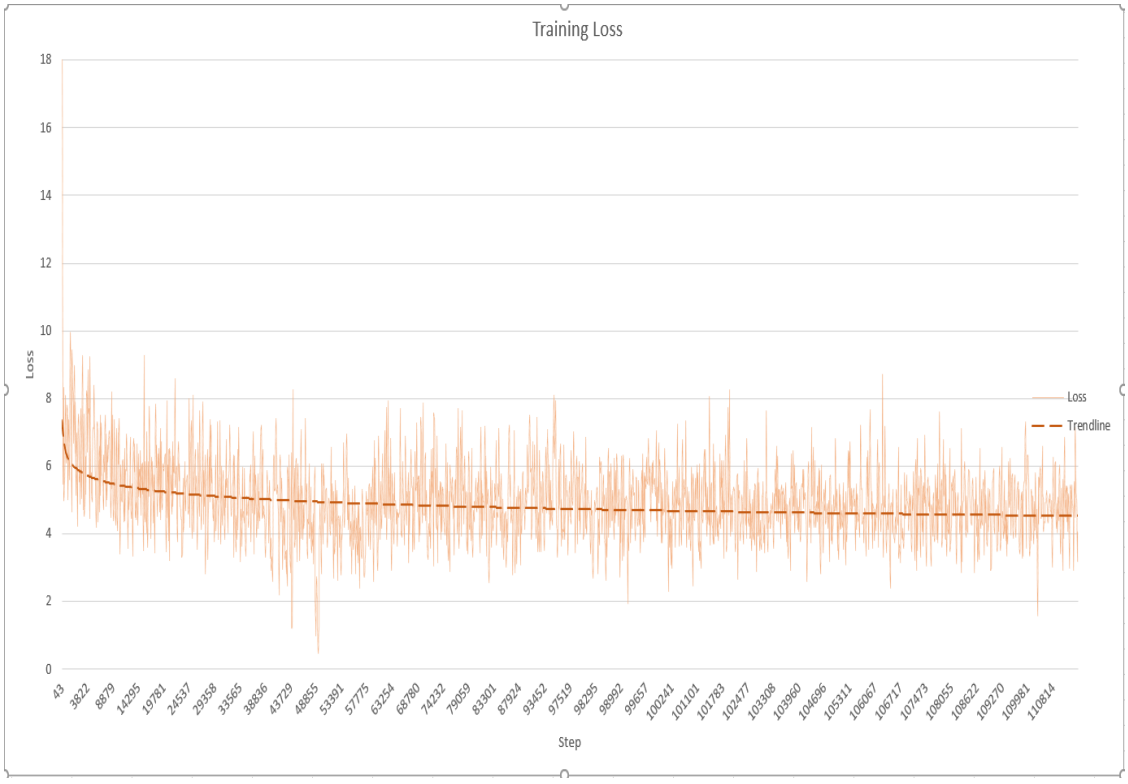


Figure 4.9 YOLO v2 training loss values through the batches

4.4.2 Testing and Results of the model

In our evaluation procedure, Yolo version 2 achieved a mAP of 44.09%. The model reached 64% of average precision on car class, 39% on cyclist and 30% on pedestrian class. A more detailed analysis of the model's performance on KITTI test set can be observed in Figure 4.10

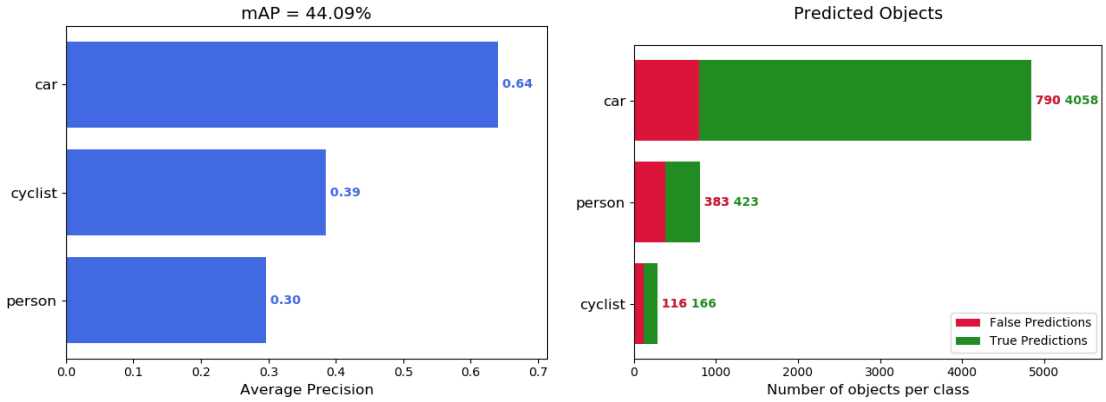


Figure 4.10 YOLOv2 evaluation results

For better overview of the results, we present the figures 4.11-4.13 showing a group of representative true positive results extracted during the model's evaluation. Green bounding boxes correspond to Yolo's predictions, blue ones correspond to the ground truths.

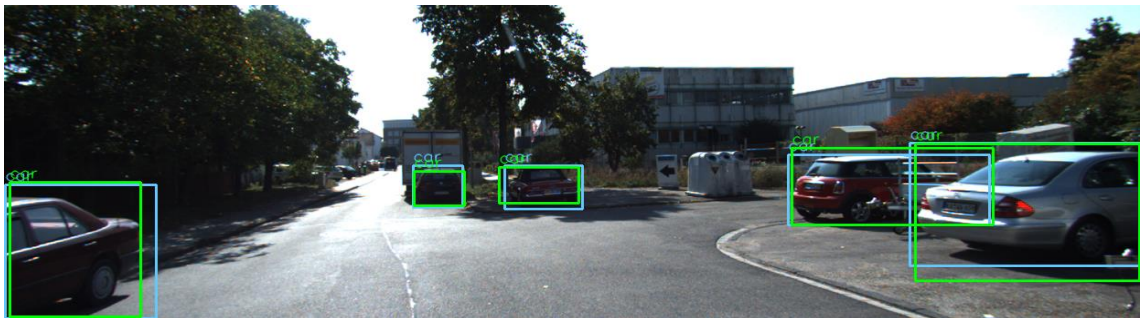


Figure 4.11. Result of Car class – Model detects all the cars located in near distance, it misses the far and small ones though (confidence threshold 0.5)

Figure 4.12. Result of Cyclist class – Moderate localization result but still a true positive detection (confidence threshold 0.5)

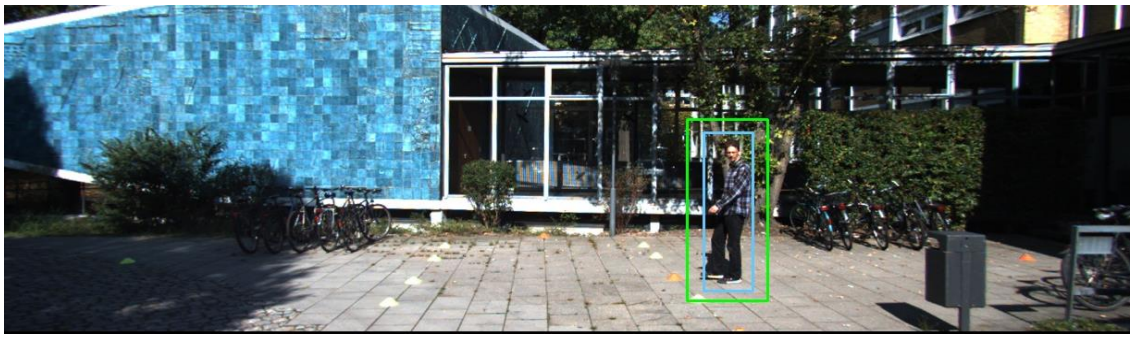


Figure 4.13. Result of Pedestrian class – Also in this sample, the position is not located with perfect accuracy, this is a true positive result though (confidence threshold 0.5)

As it is easily deduced from the evaluation metrics, YOLOv2 had more failed predictions than correct ones. Next figures (4.14-4.17) show some of these wrong estimations received during testing phase.

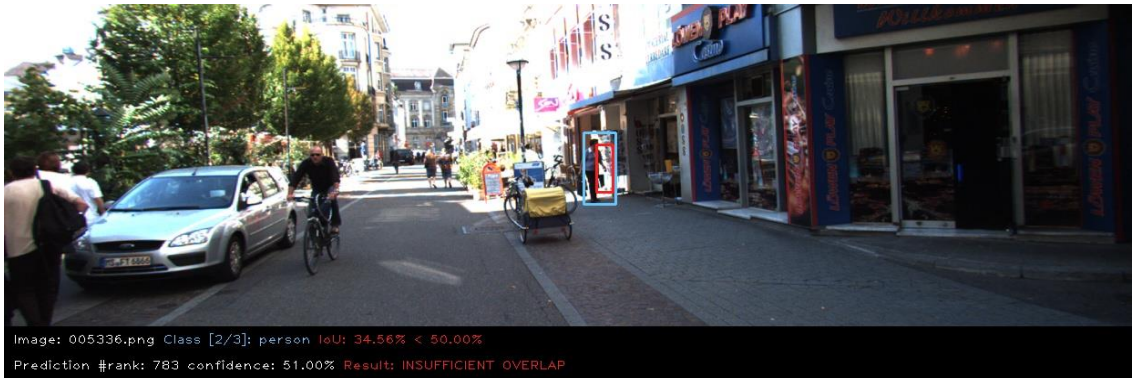


Figure 4.14. Result of Person class – The system fails to fully locate the correct position of the Pedestrian (confidence threshold 0.5).

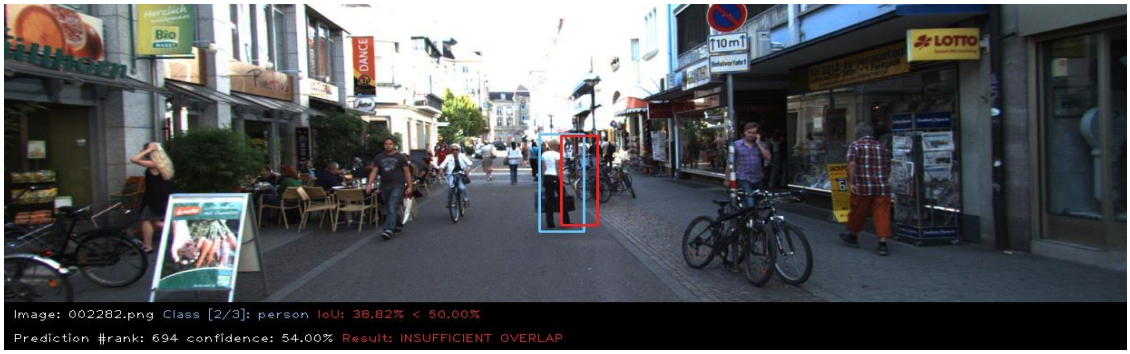


Figure 4.15. Result of Cyclist class – Similar case to the previous one, model is not able to correctly locate the position (confidence threshold 0.5).

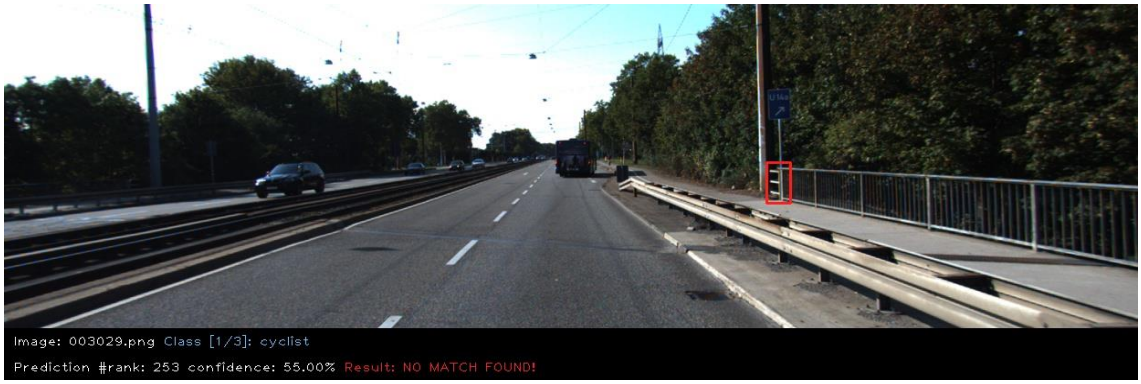


Figure 4.16. Result of Cyclist class – Railing on the edge of the road recognized as cyclist (confidence threshold 0.5).

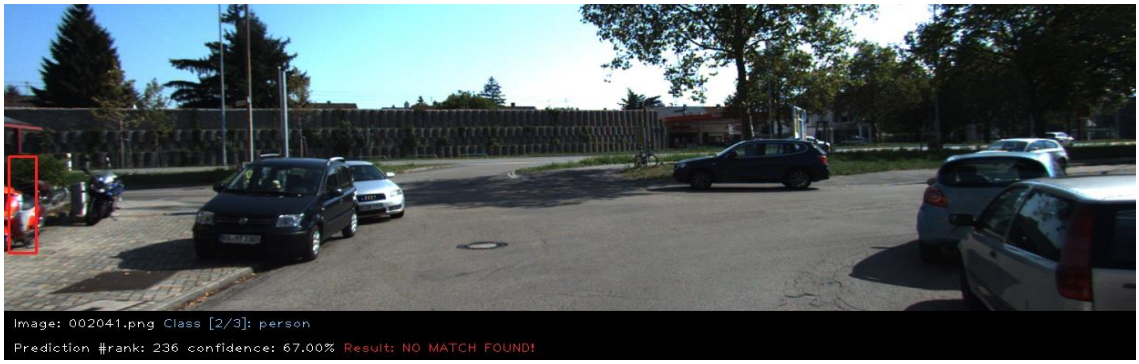


Figure 4.17. Result of Car class – Misclassification of a bicycle as a Car (confidence threshold 0.5).

4.5 YOLOv3 experiment

In our third experiment Yolo v3 was implemented. Again, main dataset was KITTI with the same train/test samples.

4.5.1 Training configuration

Pre-training of the model was performed using COCO Dataset (further information for this dataset can be found in subchapter 3.2.4). We trained the model on 160000 batches, a session that lasted about 55 hours. Each batch consisted of 16 samples. We kept the same training configuration as in 2nd version of the model so the initial learning rate was 0.001, the decay 0.0005 and the momentum 0.9.

The loss values along with their trending line during are shown in Figure 4.18.

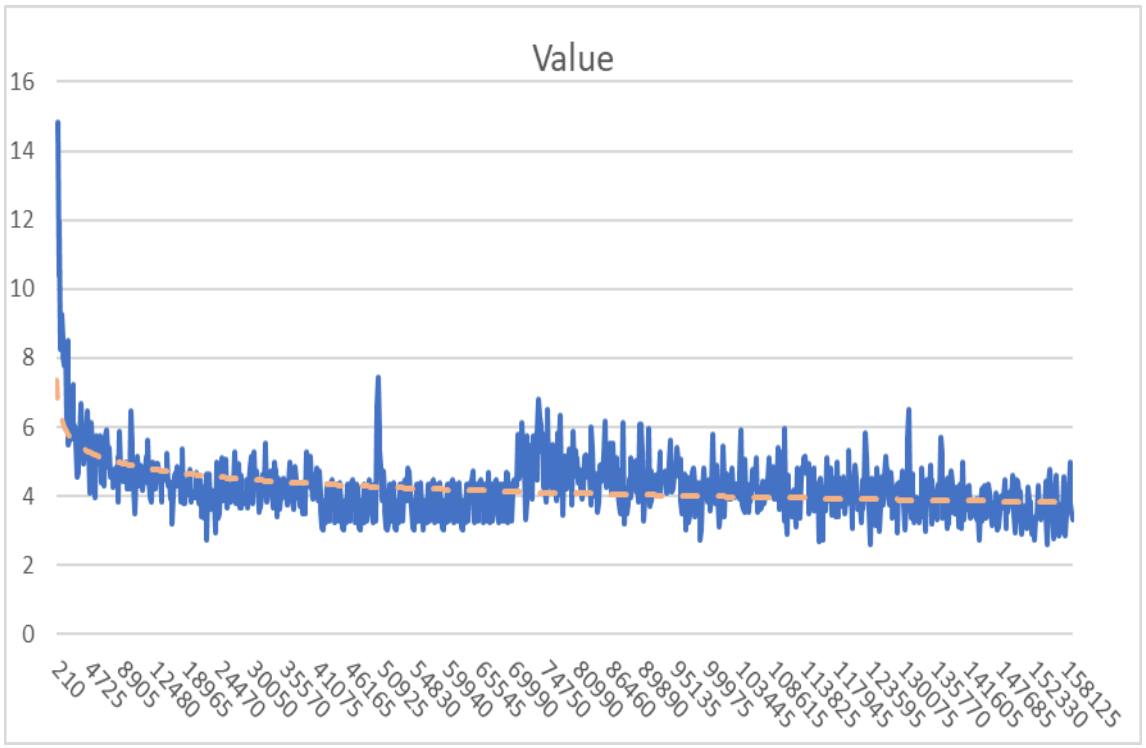


Figure 4.18 YOLO v3 training loss values through the batches

4.5.2 Testing and Results of the model

Evaluation of the third version of Yolo gave us a score of 57.52% mAP. Best result was achieved for car class with AP of 78%, while for cyclist and pedestrian classes AP was 46% and 49% respectively, as shown in Figure 4.19. A comparison of the True vs False predictions can be also found below.

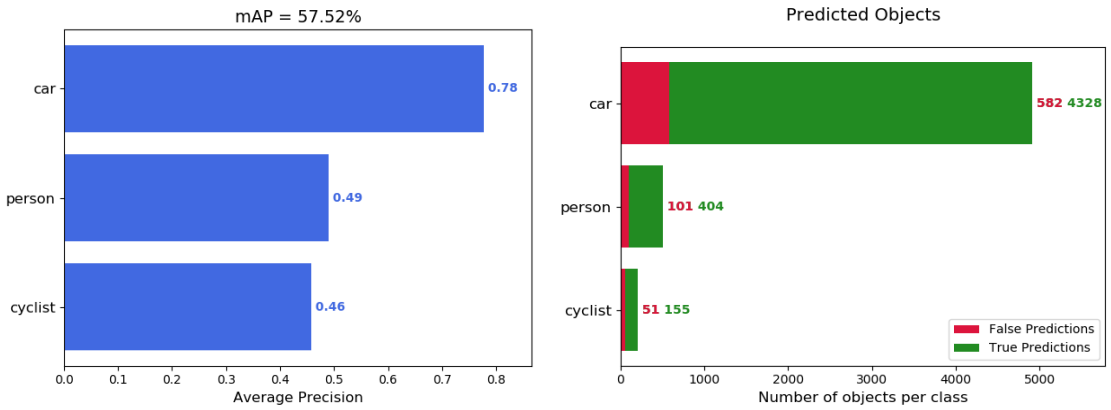


Figure 4.19 YOLOv3 evaluation results

YOLOv3 returned several good detections, especially in the car class. Following figures present some of them for a better view on the results.

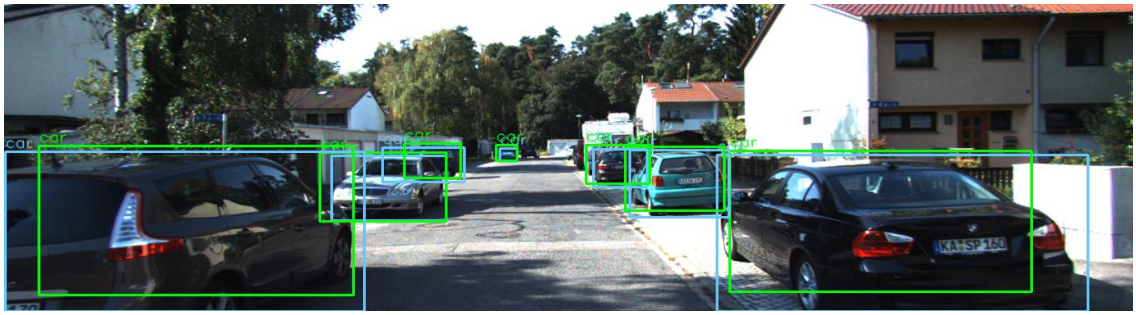


Figure 4.20. Result of Car class – Model managed to detect all cars regardless their distance from our vehicle, near or far (confidence threshold 0.5)

Figure 4.21. Result of Pedestrian class – A successful detection on the pedestrian class, ground truth and prediction boxes are very close (confidence threshold 0.5)

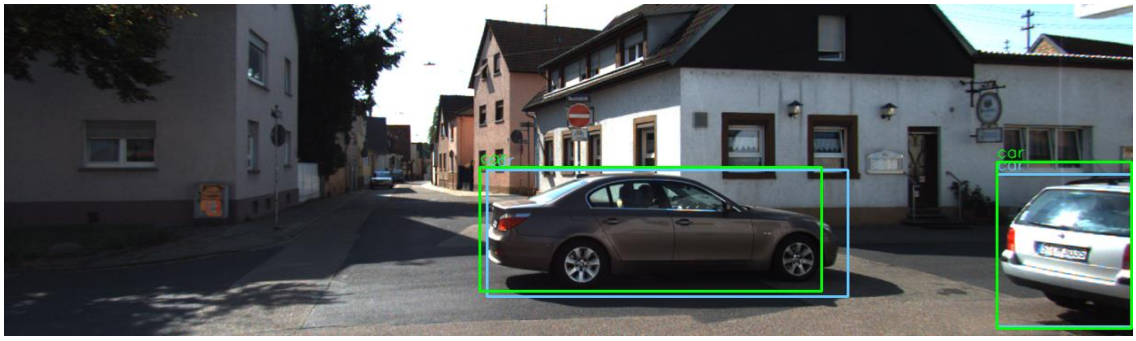


Figure 4.22. Result of Car class – In this case we have the phenomenon of occlusion. Even though the car on the right side of the image is only partially visible, model gives us a precise detection (confidence threshold 0.5)

Of course, besides the true positive results there were also noteworthy failed detections.

Following three figures reveal some weak points of Yolov3 performance on KITTI.

Figure 4.23. Result of Car class – Cyclist misclassified as a car (confidence threshold 0.5).

Figure 4.24. Result of Pedestrian class – Yolo v3 didn't manage to locate the Pedestrian correctly. Background colors probably mislead the network (confidence threshold 0.5).

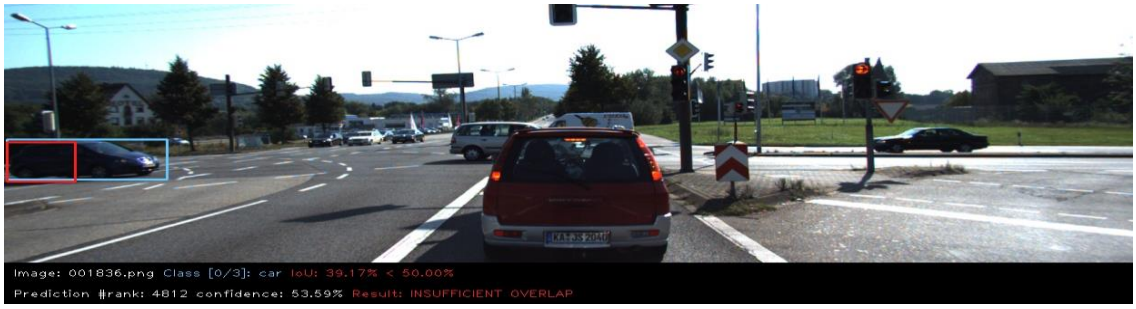


Figure 4.25. Result of Car class – Model included in the prediction only part of the car (confidence threshold 0.5).

4.6 Analysis of the results

In this section, we will interpret the results presented in subchapters 4.2 - 4.4. We will provide a comparison of the models in order to extract important conclusions.

In Table 1, we have grouped and the Average Precision results that each model scored on the three test classes. SqueezeDet achieved the best performance out of the three models with mAP of 72.54% while second best was the third version of Yolo. The worst performance was received by YOLOv2. SqueezeDet not only accomplished the best mAP but it had the greatest predictions in all three classes. In more detail, in car class it scored 4% AP more than YOLOv3 and 16% more than Yolov2. In Cyclist class it surpassed the second-best result, belonging to YOLOv3, by a remarkable percentage of 25% and YOLOv2 by 35%. In Pedestrian class SqueezeDet scored a better average precision than the 3rd version of YOLO by 16% and 32% more than the 2nd version. Above details can be also observed in Figure 4.26. All three models have scored best AP in car class, that means it is the easiest class to detect in this dataset. Worst AP was

received for Pedestrian class in all our networks thus we can conclude that is the hardest and most complex class to detect.

Model	AP Car (%)	AP Cyclist (%)	AP Pedestrian (%)	mAP (%)
SqueezeDet	82	74	62	72.54
Yolov2	64	39	30	44.09
Yolov3	78	49	46	57.52

Table 1 Average and Mean Average Precision values of the models

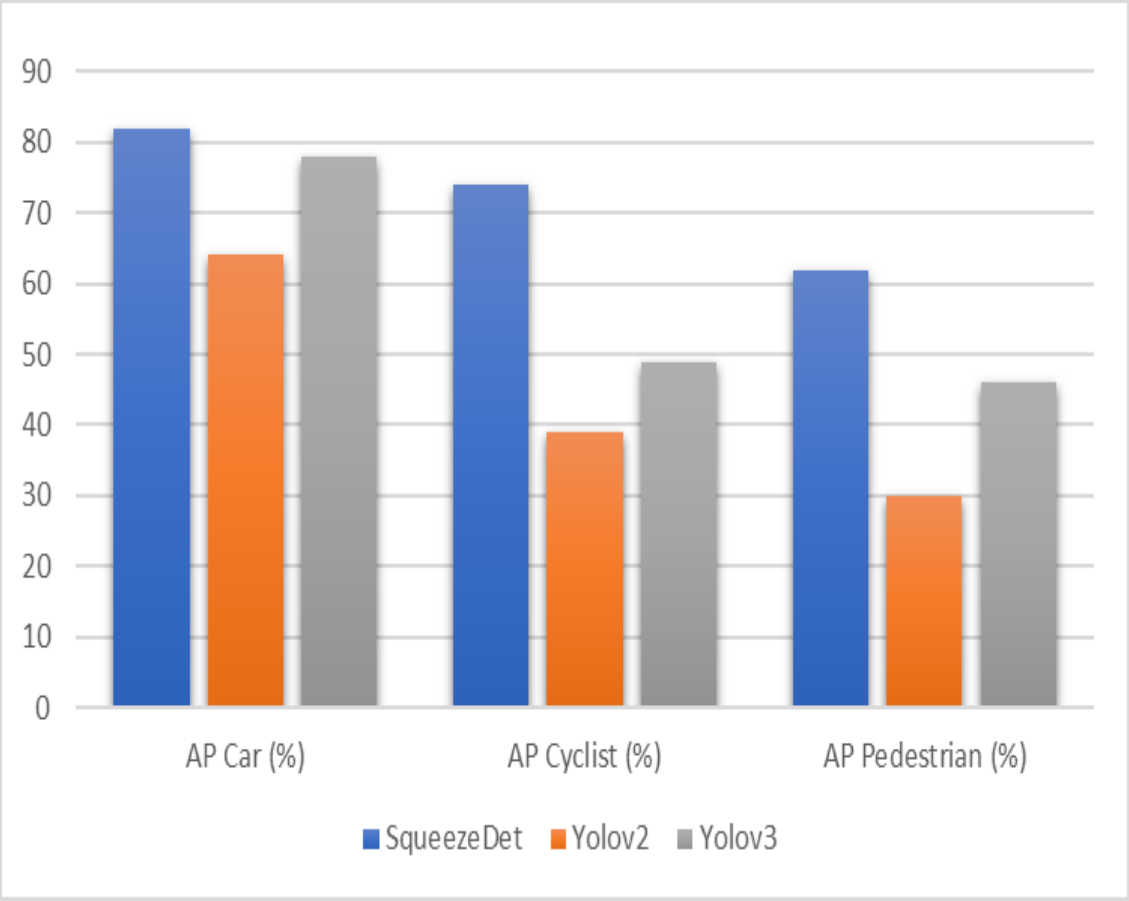
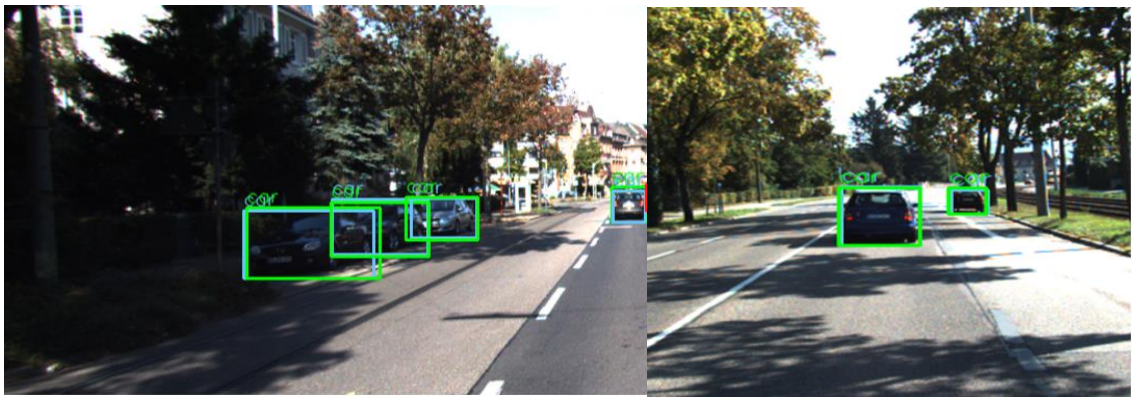


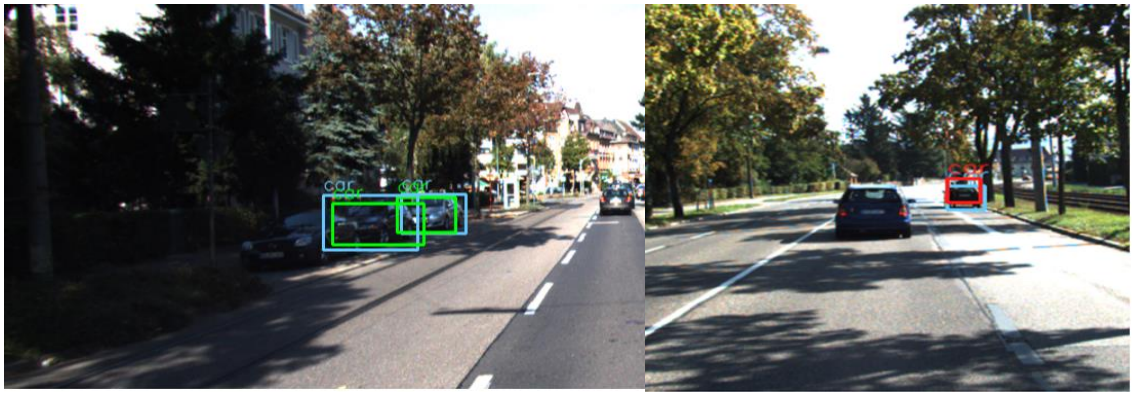
Figure 4.26 Visualization of the Table 1 - Average Precision of the models per class

In Figure 4.27, we compare all three models' detection results in two specific testing samples that include only the car class. As we can see SqueezeDet recognized accurately all 6 cars. Same result, but with slight worse localization performance, was received from Yolov3. Unfortunately, Yolov2 was able to accurately recognize only two cars,

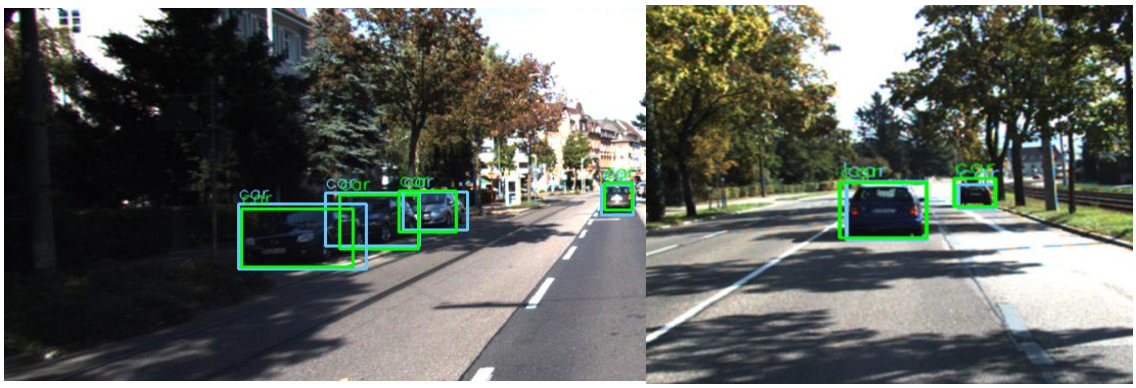
whereas three cars were not detected at all and 1 car was detected but with poor localization.



SqueezeDet



YOLOv2



YOLOv3

Figure 4.27 Visual comprehensive analysis - Car class only

In Figure 4.28, the comprehensive analysis concers Pedestrian samples only. Once again SqueezeDet did very good job. It detected with great accuracy all 3 pedestrians while the second version of Yolo failed in 2 out of 3 cases. It recognized only one pedestrian with low localization precision though. Yolov3 managed to detect all 3

cases, we can observe a poor localization precision in one case (left sample), pretty similar to Yolov2's behaviour.

SqueezeDet

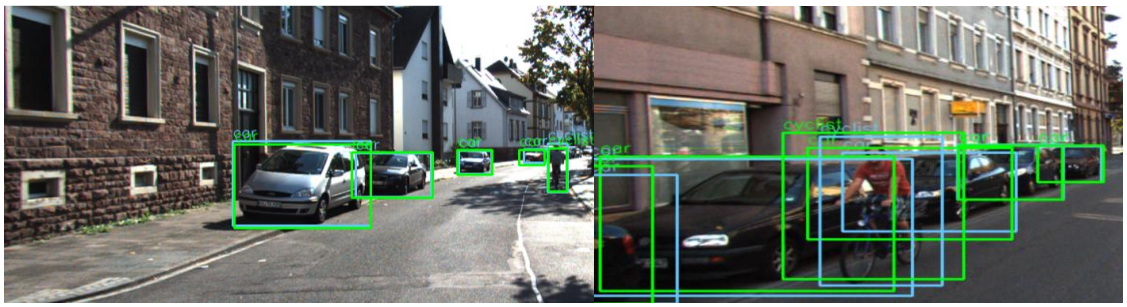
YOLOv2

YOLOv3

Figure 4.28 Visual comprehensive analysis - Pedestrian class only

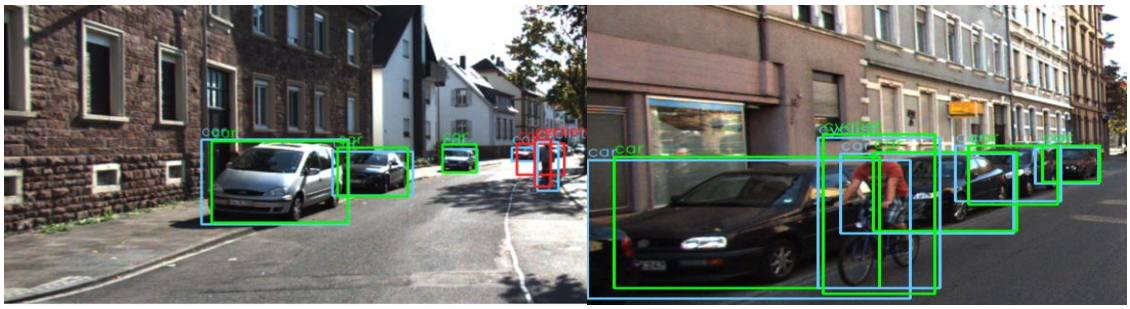
Figure 4.29 represents a more complex situation where we have combination of cars and cyclists on the road. SqueezeDet manages to detect all objects (left sample: 4 cars and a cyclist, right sample: 5 cars and 1 cyclist), even the car of the right image which only small part of its rear is visible. Yolo version 2 detects most of the cars (7 out of 9), but the car on the left image located in a far distance and also the occluded car of the

right sample were left out. Most important finding of this result though, is that Yolov2 failed to detect the two cyclists. It is worth mentioning that the model misallocated the position of the cyclist in left sample, but it totally ignored the presence of the cyclist on the right image, a mistake which could be critical on a real-life situation.



SqueezeDet

YOLOv2



YOLOv3

Figure 4.29 Visual comprehensive analysis – complex case with cars and cyclist

On the contrary Yolov3, detects accurately the cyclist on the right image and it shows a similar behavior as Yolov2 for the one on the left sample. Regarding cars' detection, Yolov3 missed the occluded car, it recognized the car in the far distance but the predicted localization was inadequate. Rest cars were accurately recognized.

5 Conclusions and future work

In the first chapter, the motivation of our research in obstacle recognition was introduced and traditional detectors along with more recent CNN-based ones were reviewed. The second chapter included a detailed review of the deep learning background focusing on the convolutional neural networks. In Chapter 3, we described the datasets and the models that were used in our experiments and in our final chapter, Chapter 4, we presented our experiments' results, followed by a comprehensive analysis.

Three different models were trained and tested in KITTI 2D detection dataset [49]. The comparison of the results has shown us some interesting findings. SqueezeDet, which was the best model with a mAP of 72.54 %, is able for real-time obstacle recognition with good precision in all three classes. It performs very well on difficult cases such as occluded obstacles and objects that are located in a far distance from our vehicle. Second best model, Yolov3, is also able for high speed inference, its detection performance is lower than SqueezeDet's though. It can recognize the cars with high accuracy, but it is not that efficient on Cyclist and Pedestrian classes. For these two classes we have observed lower localization precision compared to SqueezeDet and a difficulty to include far objects in the detection results. Yolov2 seems to be significantly weaker than the two other models. Even though it performs relatively well in Car's identification and localization tasks, there are several obvious failed detections in Cyclist and Pedestrian classes. Regarding far positioned objects, same behavior with the 3rd version was noticed. In many cases these obstacles were not identified at all. It appears that both Yolo versions do not manage to handle efficiently smaller obstacles - objects with not so "strong" features. Of course, Yolov3 performs apparently better on this task but still the outcome is not very much satisfying.

Through our experiments, we proved that SqueezeDet has great performance in image-level object detection. Thus, it makes the best choice for autonomous driving system. Yolo can also be considered a promising model for autonomous driving due to some very good aspects it showed during the tests and evaluations, but serious modifications should be applied to the current implementations in order to reach the desirable results. SqueezeDet based architectures have presented great possibility to be commercialized

due to their ability to combine high inference rates and detection results. Indeed, further research is required in order to achieve the best possible detection performance.

In this point, we should mention some restrictions we faced during our experiments. To begin with, an important obstacle was the limitations in accessible information. KITTI includes a variety of samples that greatly represent most of the cases a vehicle will face on the road, however climate (rain, snow etc.) and light conditions are not taken into consideration. Furthermore, another significant boundary was the available time and computational power for the research. Provided with more time and resources, additional experiments could be performed to the models. Moreover, besides 2D ground truth annotations, there is also 3D data information available in KITTI benchmark suite. These data could be used for the training of models in order to further investigate the capabilities of deep learning in the driving scene.

.

.

Bibliography

[1] Deng, J., Berg, A., Satheesh, S., Su, H., Khosla, A., and Fei-Fei, L. Imagenet large scale visual recognition competition 2012 (ilsvrc2012), 2012.

[2] Krizhevsky, A., Sutskever, I., and Hinton, G. ImageNet classification with deep convolutional neural networks. In NIPS, 2012.

[3] Fukushima, K. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position', Biological cybernetics, 36(4):193–202. 1980.

[4] Geiger, A., Lenz, P. and Urtasun, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on (pp. 3354-3361). IEEE, 2012.

[5] Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448, 2015.

[6] Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580–587, 2014

[7] He, K., Zhang, X., Ren, S., and Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. In European Conference on Computer Vision, Springer, pp. 346–361, 2014.

[8] Isik, E. M. Meyers, J. Z. Leibo and T. Poggio, The dynamics of invariant object recognition in the human visual system, Journal of Neurophysiology, Vol. 111 no. 1, 91-102, 2014

[9] LeCun, Y, Boser, B. Denker. J. Henderson, D., Howard, R., Hubbard, W. and Jackel, L. Backpropagation applied to handwritten zip code recognition'. Neural Computation 1, 541-551, 1989

[10] LeCun, Y. Bottou, L. Bengio, Y, and Haffner P. Gradient-based learning applied to document recognition. Proc. of the IEEE, 1998

[11] Pendleton, S., Andersen, H., Du, X., Shen, X., Meghjani, M., Eng, Y., Rus, D., and Ang, M. Perception, Planning, Control, and Coordination for Autonomous Vehicles Machines ,doi:10.3390/machines5010006 www.mdpi.com/journal/machines, 2017

[12] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788), 2016

[13] Redmon, J. and Farhadi, A. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018

[14] Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real time object detection with region proposal networks. In Advances in neural information processing systems, pp. 91–99, 2015

[15] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. Overfeat: Integrated Recognition, Localization and Detection using Convolutional Networks, 2013.

[16] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition'. arXiv preprint arXiv:1409.1556, 2014

[17] Rumelhart, D. E., Hinton, G. E., and Williams R. J. Learning internal representations by error propagation. Parallel Distributed Processing, 1:318–362, 1986.

[18] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.

[19] Uijlings, J. R., van de Sande, K. E., Gevers, T., and Smeulders, A.W. Selective search for object recognition. International journal of computer vision 104, 2, 154–171, 2013.

[20] Mikolajczyk, K., and Schmid, C. A performance evaluation of local descriptors. IEEE transactions on pattern analysis and machine intelligence 27, 10, 1615–1630. 2005

[21] Dai, J., Li, Y., He, K., and Sun, J. Object Detection via Region-based Fully Convolutional Networks, in ECCV. arXiv preprint arXiv:1605.06409, 2016

[22] Lowe, Y. Distinctive image features from scale-invariant keypoints. IJCV, 2004.

[23] Dalal, N., and Triggs, B. Histograms of oriented gradients for human detection. In CVPR, 2005.

[24] Mitchell, T. M., The discipline of machine learning. Pittsburgh: Pennsylvania, PA: Carnegie Mellon University, 2016.

[25] Mohri, M., Rostamizadeh, A., Talwalkar, A. Foundations of Machine Learning, The MIT Press ISBN 9780262018258, 2012.

[26] Bengio, Y.; LeCun, Y., and Hinton, G. "Deep Learning". Nature. 521 (7553): 436–444. Bibcode:2015Natur.521..436L. doi:10.1038/nature14539. PMID 26017442, 2015

[27] Oliveira, R. M. S. de, Araújo, R. C. F., Barros, F. J. B., Segundo, A. P., Zampolo, R. F., Fonseca, W., Dmitriev, V., and Brasil, F. S. A System Based on Artificial Neural Networks for Automatic Classification of Hydro-generator Stator Windings Partial Discharges. Journal of Microwaves, Optoelectronics and Electromagnetic Applications, 16(3), 628-645. https://dx.doi.org/10.1590/2179-10742017v16i3854 , 2017.

[28] Bishop, C. M. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[29] Hubel, D. H., and Wiesel, T. N. Receptive fields and functional architecture of monkey striate cortex. The Journal of Physiology 195, 1 1968.

[30] Marr, D., and Hildreth, E. Theory of edge detection. Proceedings of the Royal Society of London B: Biological Sciences 207, 1167, 1980.

[31] Moncada S. CONVOLUTIONAL NEURAL NETWORKS (CNN): STEP 1- CONVOLUTION OPERATION https://www.superdatascience.com/convolutional-neural-networks-cnn-step-1-convolution-operation/ (Accessed 5/12/2018).

[32] Karpathy A. CS231n Convolutional Neural Networks for Visual Recognition. Convolutional Neurons. http://cs231n.github.io/convolutional-networks/ (Accessed 1/12/2018).

[33] Karpathy A. CS231n Convolutional Neural Networks for Visual Recognition. Neural Networks Part 1. http://cs231n.github.io/neural-networks-1/ (Acessed 19/9/2018).

[34] Yu, D., Wang, H., and Chen, P. Mixed pooling for convolutional neural networks. Int. Conf. Rough Sets and Knowledge Technology, Shanghai, China, pp. 364–375, 2014.

[35] Prabhu R., Understanding of Convolutional Neural Network (CNN)—Deep Learning. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148 (Accessed 1/12/2018)

[36] Deshpande, A., A Beginner's Guide To Understanding Convolutional Neural Networks. https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/ (Accessed 8/11/2018)

[37] Wu, B., Iandola, F., Wan, A., Jin, P. H., and Keutzer, K. SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving, in CVPR, 2017.

[38] Christopher, E., Fast object detection with SqueezeDet on Keras https://medium.com/omnius/fast-object-detection-with-squeezedet-on-keras-5cdd124b46ce (Accessed 9/11/2018)

[39] Shikhkerimov, O., Mishchenko, I., Real-time Object Detection with Convolutional Neural Networks. https://aiukraine.com/wp-content/uploads/2017/10/1_5-Mishchenko-Shikhkerimov.pdf (Accessed 28/10/2018)

[40] Redmon, J. and Farhadi, A., (2016) Yolo9000: A Better, Faster, Stronger. arXiv preprint arXiv:1612.08242.

[41] Sergey, I., Christian, S., Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv preprint arXiv:1502.03167, 2015.

[42] Zingade, A. Logo detection using YOLOv2, https://medium.com/@akarshzingade/logo-detection-using-yolov2-8cda5a68740e (Accessed 10/9/2018)

[43] Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2117–2125, 2017.

[44] Russakovsky, O., Deng, J., Berg, A., Satheesh, S., Su, H., Khosla, A., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. arXiv preprint arXiv:1409.0575. 2014.

[45] Everingham, M., and Winn, J. The pascal visual object classes challenge 2011 (voc2011) development kit. Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep., 2011.

[46] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL Visual Object Classes (VOC) Challenge, 2007.

[47] Lin,T.-Y., Maire, M., Belongie, S.,Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár. Microsoft COCO: Common Objects in Context. arXiv preprint arXiv:1405.0312, 2015.

[48] COCO Explorer, http://cocodataset.org/#explore (Accessed 12/08/2018)

[49] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. Vision meets robotics: The kitti dataset. The International Journal of Robotics Research, 0278364913491297, 2013.

[50] Russell, S. J., and Norvig P. Artificial Intelligence: A Modern Approach, Third Edition, Prentice Hall ISBN 9780136042594, 2010.

[51] A Closer Look at YOLOv3, https://www.cyberailab.com/home/a-closer-look-at-yolov3 (Accessed 12/08/2018)

[52] TensorFlow. https://www.tensorflow.org/ (Accessed 9/08/2018)

[53] Scanlan D., and Solana L. D., Deep Learning for Robust Road Object Detection Using Convolutional Neural Networks to Detect Cars, Pedestrians, & Cyclists in Single Colour Images, Master Thesis, Chalmers University of Technology, 2017.

[54] Sonawane A., YOLOv3: A Huge Improvement (Accessed 13/11/2018) https://medium.com/@anand_sonawane/yolo3-a-huge-improvement-2bc4e6fc44c5

[55] Colyer A. ImageNet Classification with Deep Convolutional Neural Networks. https://blog.acolyer.org/2016/04/20/imagenet-classification-with-deep-convolutional-neural-networks/ (Accessed 29/11/2018)

[56] Novotny, D., Large scale object detection, Master Thesis, Czech Technical University in Prague, 2015.

Appendix

Table of figures